information

Article

Extending the Hierarchy of System Specifications and Morphisms with SES Abstraction

Bernard P. Zeigler [1,2]

[1] Department of Electrical and Computer Engineering, University of Arizona, Tucson, AZ 85721, USA; zeigler@rtsync.com
[2] RTSync Corp., Chandler, AZ 85226, USA

Abstract: This article works toward a unification of two related concepts that underpin system-theory-based modeling and simulation–the hierarchy of system specifications and morphisms and the System Entity Structure (SES). The hierarchy organizes system specification along levels ranging from behavior to structure capturing increasing knowledge of the system input/output processing and state dynamics. The SES is a constructive ontology describing compositions of modular components via coupling of input/output ports. Toward unification of these concepts, we propose an abstraction of the SES called the MetaSES that supports the construction of complex systems of systems with multiple components belonging to specified classes. Moreover, we place the MetaSES within a computational framework with the goal of making it easier to design and build complex hierarchical DEVS models and to communicate their structures and intended behaviors to foster continued reuse and development. We discuss several examples of applications to illustrate how the MetaSES-based enhancement of the hierarchy of system specifications and morphisms helps to push the boundaries of complexity management in the theory and practice of modeling and simulation. Research directions stemming from the proposed concepts are suggested.

Keywords: modeling and simulation; hierarchy of system specifications; system entity structure; DEVS; discrete event system specification; metamodeling

Citation: Zeigler, B.P. Extending the Hierarchy of System Specifications and Morphisms with SES Abstraction. *Information* **2023**, *14*, 22. https://doi.org/10.3390/info14010022

Academic Editor: Vladimír Bureš

Received: 8 November 2022
Revised: 26 December 2022
Accepted: 27 December 2022
Published: 29 December 2022

1. Introduction

This article appears in the context of the Open Special Issue of Information Journal called "Simulation Modeling Theory and Practice: Pushing the Boundaries". The article addresses the challenge of pushing the boundaries by extending the hierarchy of system specifications and morphisms which has been formulated as a rigorous system-theoretic foundation for modeling and simulation (M&S) theory and practice [1]. For readers who are not familiar with the current state-of-the-art of M&S it is instructive to quote from the solicitation for the special issue as written by the editors:

> "In recent decades, modeling and simulation has evolved to the point where it is now a recognized discipline and provides infrastructure for disciplines spanning the whole spectrum of human knowledge, intellectual and practical effects. The term "modeling and simulation" grants equal stress on both modeling and simulation aspects of this discipline. The Modeling and Simulation Body of Knowledge (M&S BoK) is a living work in process with the goal of establishing a kernel of topics that categorically characterize the discipline of modeling and simulation as it drives the progress of multifold disciplines in science, engineering, and the arts. The essence of M&S is the creation of conceptual representations of entities, relations, and processes describing a problem domain with the goal of making them computationally executable artifacts. Such simulation models can be used to perform experiments or to gain experience; both dimensions provide a multitude of possibilities".

Within this broad context, we note that the development of families of coherent representations of real-world phenomena (climate change, pandemics, global food supply, etc.,) are needed to support the computational exploration of simulations ranging from spreadsheet analysis to high-performance cloud-based execution [1]. We contend that the long-term development of such families should be founded on a firm system theory basis informed by model structures leveraging computational resources. Zeigler [2] proposed a methodology based on the theory of modeling and simulation [3] founded on system-theoretical concepts of Wymore [4] and others. In this article, we attempt to unify two related concepts that underpin system-theory-based M&S–the hierarchy of system specifications and morphisms and the System Entity Structure (SES). The hierarchy organizes system specification along levels ranging from behavior to structure capturing increasing knowledge of the system input/output processing and state dynamics. The SES is a constructive ontology describing compositions of modular components via coupling of input/output ports. Toward unification of these concepts, we propose an abstraction of the SES called the MetaSES that supports construction of complex systems of systems with multiple components belonging to specified classes. Moreover, we place the MetaSES within a computational framework with the goal of making it easier to design and build complex hierarchical DEVS models and to communicate their structures and intended behaviors to foster continued reuse and development.

The underlying methodology [2] critically employs the hierarchy of system specifications and morphisms as outlined:

- **Use of system specification hierarchy and associated morphisms.** The levels of system specification range from lowest level behavior specification to highest level structural specification [3]. Corresponding to each system specification level is a morphic relation appropriate to a pair of systems specified at that level. Morphisms at each level are defined such that a morphism which preserves the structural features of one system in another system at one level also preserves its features at all lower levels.
- **Development Method**. The morphisms at the Input/output Function and State Transition levels underlie the minimal realization and homomorphic image concepts supporting the quest for minimal explanatory forms and computationally feasible implementations. Validation of the latter proceeds via proofs and Discrete Event System Specification (DEVS)-based simulations. The prioritization of behaviors for first consideration is motivated by the desire to come up with, and define, building blocks and architectural coupling patterns for ubiquitous, composable, and reusable application [2].
- **Minimal forms**. In line with the hoary dictum of philosophy, Occam's razor, we seek explanations of behavior that contain only those assumptions that are necessary to the explanation. However, the minimal realizations that we seek are based on concepts formulated in mathematical systems theory derived from both linear systems theory and finite automata theory [5]. Proving that a realization of a behavior is minimal in this sense implies that it is a homomorphic image in relation to any implementation of the same behavior. Moreover, definitions for state-based realization of behaviors based on mathematical system theory and DEVS fundamentally include temporal and probabilistic characteristics of system inputs, state, and outputs [3]. Moreover, they provide a solid system-theoretical foundation and simulation modeling framework for both low and high-performance computational support of complex phenomena model development.
- **Network construction**. The hierarchy of system specifications includes levels for definition of networks of components with coupling specification. This is exemplified by the DEVS coupled model definition with its well defined coupling specification. The proof of closure under coupling shows how resultant networks are equivalent to basic models, and can be treated as such in hierarchical construction [3].
- **Model formalism for Simulation and Design.** DEVS enables formal and complete description of hybrid continuous/discrete model components and subsystems. DEVS-

based software tool sets provide atomic model and hierarchical coupled model specifications that support graphical description of the internals and interfaces of component behavior combining energy, material, and information flows. The hybrid DEV&DESS [3] formalisms enable expressing differential and algebraic equations for energy-related internal variables intermixed with discrete behavior described in state-based system form. Finally, transparent implementation of the canonical DEVS abstract simulator for handling events and equations enable design of dedicated simulation functionality.

Zeigler [2] provides a detailed comparison with some prominent methodologies for development of models of mind [6–8]. Although such methodologies support comparable aspects of the above dimensions, they fail to integrate within their frameworks the computational unification for simulation provided by the hierarchy of systems specifications and morphisms. More specifically, as we shall illustrate, multiple levels of the hierarchy may be simultaneously at play in modeling or design necessitating the ability to conveniently span the hierarchy levels. As Wilsdoft et al. [9] state, "Providing a structured representation of the various ingredients of simulation experiments in the form of metamodels and collecting them in a repository improves knowledge sharing across application domains and simulation approaches". The framework discussed here provides a formal system theoretically grounded, and computational justified, approach to developing the metamodels in a coherent and integrated manner.

Table 1 informally reviews the levels of system specification ranging from lowest level behavior specification to highest level structural specification. Corresponding to each system specification level is a morphic relation appropriate to a pair of systems specified at that level. Morphisms at each level are defined such that a morphism which preserves the structural features of one system in another system at one level also preserves its features at all lower levels [3].The morphisms at the I/O Function and State Transition levels are the ones that underlie the minimal realization and homomorphic image concepts supporting the quest for minimal forms mentioned earlier. Examples are presented in the table that relate to the applications of the methodology to be discussed in the sequel.

Table 1. Informal Description of the Levels of System Specification.

Level	Specification Name	What We Know at This Level	Examples to Be Discussed in the Sequel
0	I/O Frame	How to stimulate the system with inputs; what variables to measure and how to observe them over a time base;	Input: set of symbols from an alphabet Output: Boolean variable indicating acceptance or not of input stream as sequence in regular language
1	I/O Relation	Time-indexed data collected from a source system; consists of input/output pairs	Union of I/O Functions over all initial states so that multiple output segments may result from the same input segment (depending on which state actually was in force initially.)
2	I/O Function	Knowledge of initial state; given an initial state, every input segment produces a unique output segment.	Pairing of unique output value with input segment of symbols indicating acceptance, or not, of input stream as sequence in regular language.
3	I/O System	How states are affected by inputs; given a state and an input what is the state after the input stimulus is over; what output event is generated by a state.	This is the specification level at which the minimal realization of an I/O Function resides. For example, the acceptor with the smallest number of states for a given language is described at this level.

Table 1. *Cont.*

Level	Specification Name	What We Know at This Level	Examples to Be Discussed in the Sequel
4	Structured System	The I/O System state is described in terms of a cross-product of state sets, such as a point in a vector space.	The description of the Random Forest model as it transits from global state to global state under input and activation events.
5	Multi-component System	The system is specified as composition of components whose outputs are directly linked to inputs of other components	The Verilog implementation of the Random Forest model resides at this level where components are tightly "wired" lacking the ability to stand alone.
6	Network of Systems	Components and how they are coupled together. The components can be specified at lower levels or can even be structure systems themselves–leading to hierarchical structure.	The SES-based implementation of the Random Forest model resides at this level as a composition of modular systems with explicit coupling rather than hard wired connections.

2. Operationalizing the Hierarchy of System Specifications and Morphisms

Figure 1 illustrates a conceptual framework proposed to operationalize the hierarchy of system specifications and morphisms. MetaSES is a meta-level abstraction of the components and couplings of the System Entity Structure (SES) [10] that provides a computational ontology to enhance modeling and simulation at the coupled model level of the hierarchy. More explicitly, a MetaSES is an abstract specification of an SES that recognizes classes of DEVS models as components and provides a high-level description of the port-to-port coupling that routes message flow among such components. As such, a MetaSES leverages the SES with concepts and tools that refine and extend existing ones to enhance support for creating SESs and thereby exploit its ability to be transformed into simulateable DEVS coupled models [11,12].

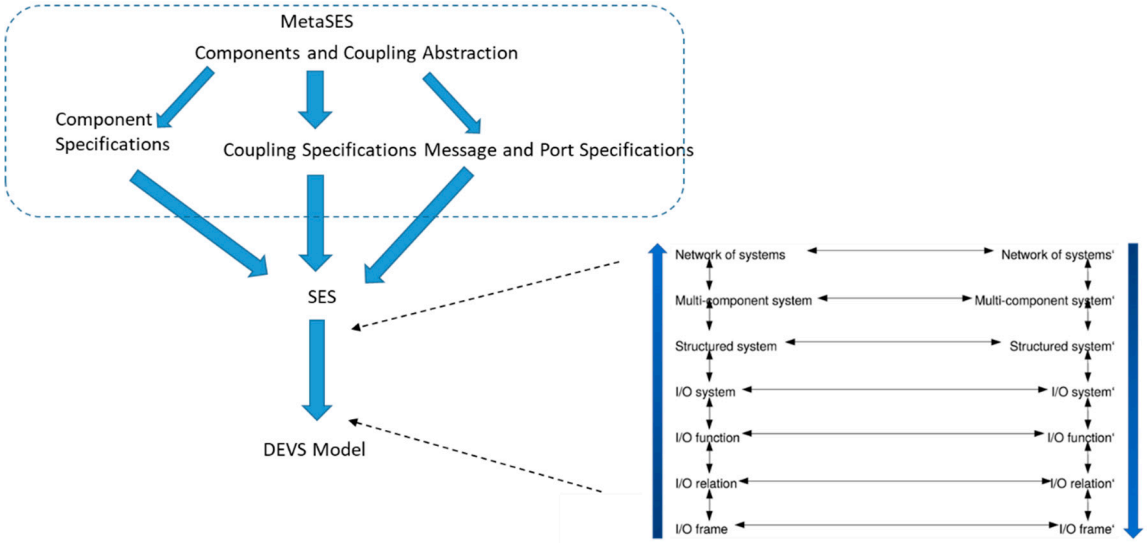

Figure 1. MetaSES framework operationalize the hierarchy of system specifications and morphisms.

Before proceeding to more detailed exposition, an example from Figure 6.4 in [13] is instructive. That SES considers a Dating Game made up of a group of boys and a group of girls, in which all boys say hello to each other, all girls say hello to each other, and girls greet boys. A MetaSES abstraction of this SES would specify a pair of model classes (B,G) and couplings: All B send Hello to all B, all G send Hello to all G, and all G send Greeting to all B, as specified in constrained language form patterned after that for the SES described in [13]. As shown in that example, with respective models resident in a model repository for Boy and Girl components that can consume and transmit messages labeled Hello and

Greeting, the SES can be automatically transformed into a simulateable model called the Dating Game.

The MetaSES is an abstract specification of an SES that specifies classes of components and a high-level description of the port-to-port coupling. Clearly, it needs to a lot of elaboration to provide enough detail for a DEVS simulation. As shown in Figure 1, elaboration can involve components (typically specified at the lower levels of the hierarchy), coupling specifications (at the network of systems or coupled model levels) and message/port specifications (spanning component and network levels). Nevertheless, the information specified by the MetaSES is critical to a top down construction process and offers a standalone abstraction that can unify separately developed models into a unified family. Moreover, it provides an initial pattern from which to develop such a family through the elaboration process to be discussed next.

We illustrate this process in Figure 2. The MetaSES generates a labelled directed graph (digraph). This induces a labelled digraph generated by an unrestricted SES that is its digraph morphic pre-image. Sub-digraphs of the latter result from further imposition of constraints resulting in transformable SESs to be simulated in computational experiments. An example of a class of neural nets illustrates this idea: The MetaSES on the lower right specifies that such neural nets consist of classes of Neurons and Synapses which exchange spikes (all Neurons send Spike to all Synapses, all Synapses send Spike to all Neurons). The unrestricted digraph at the upper right specifies individual neurons and individual synapses with all-to-all coupling connecting them (and no internal couplings within these groups). We call this the *free* digraph pre-image generated by the MetaSES. Finally, the SES at the bottom left eliminates most couplings so that only specified (neuron, synapse) and (synapse, neuron) pairs are coupled.

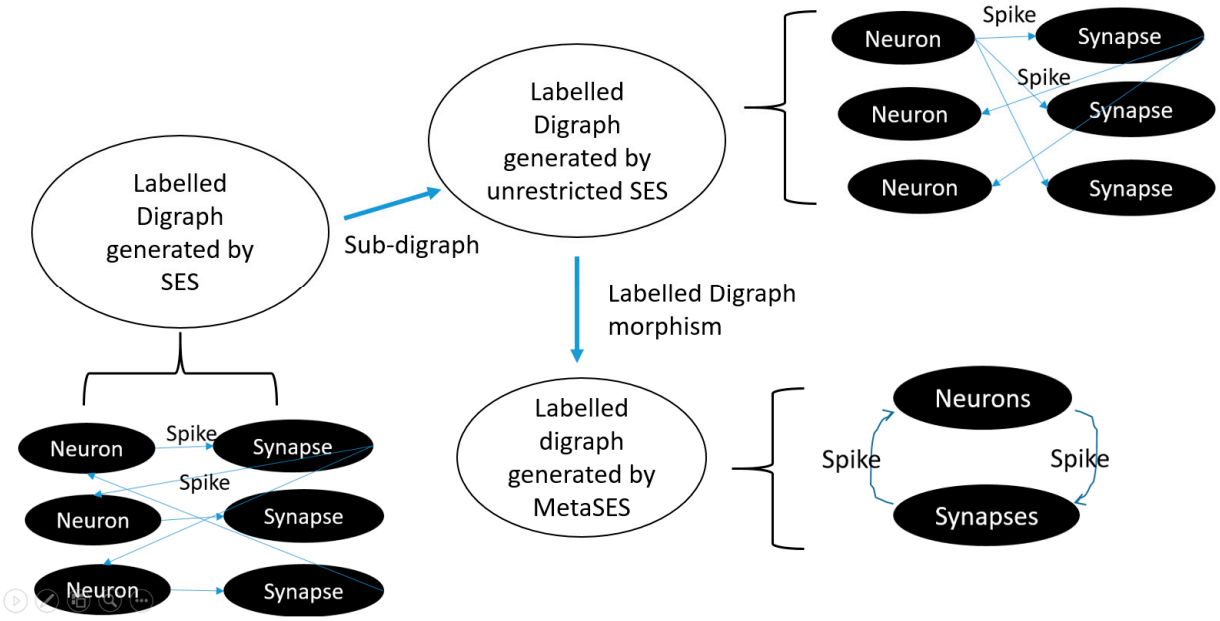

Figure 2. Illustrating the process of elaborating an MetaSES to construct a DEVS model.

The discussion above is summarized succinctly as:

MetaSES for Neural Net:

- *Classes*: N, S
- *Coupling*: N sends Spike to S, S sends Spike to N

Free Digraph pre-image of MetaSES:

- *Components*: neurons in N, synapses in S
- *Coupling*: all-to-all N to S via Spike, all-to-all S to N via Spike

Transformable SES:

- *Components*: neurons in N indexed 1,2, . . . |N|, synapses in S indexed by all pairs (i,j) where i ! = j
- *Coupling*: each neuron sends Spike to the synapses with the same left index,

  each synapse sends Spike to the neuron with the same right index

Note that the last version specifies the usual pattern where the synapse indexed by (i,j) represents the effect of neuron i's output on neuron j's input.

Thus the concept of MetaSES can be seen as implemented in three stages: *First*, the classes of components and the types of messages exchanged among them are laid out; *second*, the implied components and couplings are generated; and *third*, a variety of potential system entity structures can be generated by placing more constraints on the component numbers and their couplings. Note that stage 2 is algorithmic and can be automated while stage 3 can be supported by suitable tools, thus opening up the potential for exploration of much more complex spaces than previously possible. Appendix A.1 sketches an algorithm for performing stage 2 which has been implemented in MS4 Me (Seo et al. 2013). In the sequel, we illustrate the concept with variations that arise in specific applications: Artificial Neural Nets, Language Acceptance and Generation, Dynamic Neuronal Ensembles and Random Forests of Decision Trees. A summary of the notable features of each class evinced by the MetaSES approach appears below. Operations on MetaSESs such as component manipulation, coupling modification, and message elaboration are illustrated in the examples to be discussed.

Summary of MetaSES Application Examples

Table 2 provides a summary of the examples to be discussed that offers a perspective on the range of complex model descriptions that the MetaSES can support.

Table 2. Summary of MetaSES application examples to be discussed.

Type of Network Model	Hierarchy	MetaSES	Transformable SES
Artificial Neural Nets	Single level	Neurons and Synapses are coupled via Spikes	Synapses are in one-one correspondence with pairs of Neurons
Language Acceptance	Single level	Activation Nodes and Elementary Perception Units are coupled by Activate signals to recognize external input streams of symbols.	Activation and input have distinct coupling patterns; DeActivateable nodes enable later solutions to win.
Language Generation	Single level	Activation Nodes and Elementary Generation Units are coupled by Activate signals to generate external output streams of symbols.	Activation and output have distinct coupling patterns; Competing transitions vie for activation with a bidding protocol.
Dynamic Neuronal Ensembles	Two level composition of (1) Neurons and (2) Neuron components	(1) Neurons at the top level of a hierarchical composition are coupled via Spikes. (2) Neurons are composed of dendrites, cellbody interacting via fire signals and Spikes.	Axon outputs and dendrite inputs are individual exposed as ports at the Neuron level to enable control of axon to dendrite couplings.
Random Forest of Decision Trees	Two level composition of (1) Decision trees and (2) Decision tree components	(1) Forest composed of trees in parallel composition with output collector. (2) Decision Tree is composed of input image, Yes/No Decision models and report nodes.	Yes/No Decision models, receive external features from the input image and channel decision making via downstream Activate signals to final classification report notes.

3. Application to Event-Based Language Processing

Application of the MetaSES paradigm to finite state language generation and recognition will show its utility. Figure 3 illustrates a workflow that starts with a Finite State Automaton (FSA) description which is mapped through all three stages just mentioned into a MetaSES and then into an SES and finally, a DEVS coupled model that can be simulated in software and/or implemented in hardware.

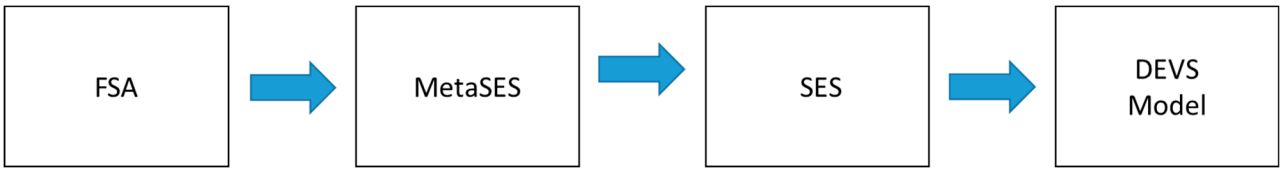

Figure 3. Mapping chain from a Finite State Automaton (FSA) into a MetaSES, SES and DEVS coupled model.

The example of language acceptance/generation will illustrate how the MetaSES concept enables abstract representation of the common structural features of a wide family of models (regular language devices).

3.1. Finite State Language Acceptance

Based on background on DEVS representation of language acceptance and generation in [14] consider the following MetaSES construction:

$$\text{Given FSA} = < Q,X,\text{delta},q_o,F>$$

where Q is a finite set of states,
X is an alphabet
Delta is a partial mapping from QxX to Q
q_o is the initial state
F is a subset of Q, the final states, define the MetaSES as follows:

- *Classes*: AN (Activation Node), Elementary Perception Unit (EPU), First Arrival (FA)
- *Coupling*:
  - FSA sends X to EPU, (external input coupling)
  - EPU sends Activate to AN, (internal coupling)
  - AN sends Activate to EPU,
  - AN sends Activate to FA,
  - FA sends Activate to FSA (external output coupling)

As depicted in Figure 4, the coupling pattern makes clear the types of components and the interaction among them as mediated by the types of messages. We can see that external alphabet inputs X arrive to the EPUs separately from the activation arrivals (this is distinct from typical artificial neural nets (ANN) in which data arrival serves as both stimulus and activation. Further the external output is an Activate message that is produced when a DEVS input segment is recognized. When incorporated into a larger coupled model, this output can serve to activate other downstream components such as decision elements. Although this pattern is fairly simplistic, the utility of this representation will become more apparent as additional features are introduced to handle more complex requirements.

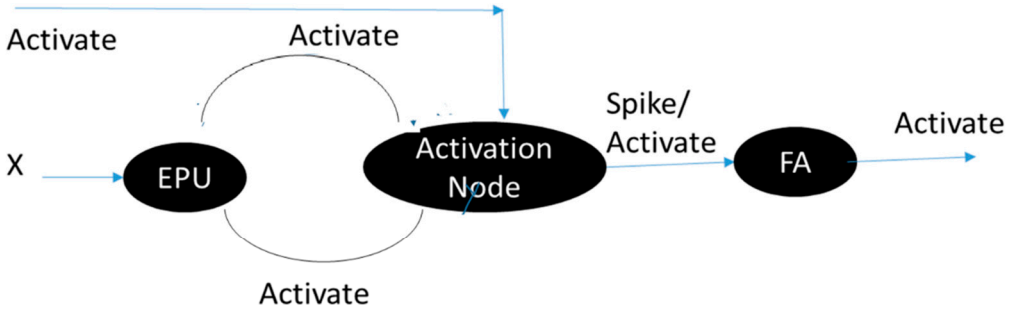

Figure 4. FSA coupling pattern showing the types of components and their interaction.

Carrying out the subsequent stages of the process in Figure 2, we obtain the MetaSES and the Transformable SES:

Free Digraph pre-image of MetaSES:

- *Components*:
  - $\{AN_q \mid q \in Q\}$ (Activation Node for each state)
  - $\{EPU_x \mid x \in X\}$ (EPU for each input element}
  - FA
- *Coupling*:
  - FSA sends Activate to all AN (external input coupling)
  - FSA sends X to all EPU (external input coupling)
  - all EPU sends Activate to all AN,
  - all AN sends Activate to all EPU,
  - all AN sends Activate to FA, (internal coupling)
  - FA sends Activate to FSA (external output coupling)

Transformable SES:

- *Components*:
  - $\{AN_q \mid q \in Q\}$ (Activation Node for each state)
  - $\{EPU_{x,i} \mid x \in X, i = 1, \ldots,$ Number occurrences of x in delta} (There is one EPU for each occurrence of x, and it causes a distinct state transition so it must be identified uniquely)
  - FA
- *Coupling*:
  - FSA sends Activate to AN_{q0}
  - For each q in F, AN_q sends Activate to FA,
  - FA sends Activate to FSA
  - FSA sends X to all EPU

Of the EPUs that recognize the input symbol, the one that has been last activated will be the one to respond and activate the successor nodes–we describe the coupling to implement this requirement in the following algorithm sketch:

Algorithm Sketch:
Assignment algorithm to properly restrict couplings of the form: all EPU sends Activate to all AN, and all AN sends Activate to all EPU.
For each $x \in X$, index the occurrences of x
Define Dict = $\{(q,x_i,q')$ where $q' = delta(q,x_i)$ and x_i is the ith occurrence of $x \in FSA\}$
For each triple $(q,x,q') \in$ Dict

Add couple: AN_q sends Activate to $EPU_{x,i}$
Add couple: $EPU_{x,i}$ sends Activate to $AN_{q'}$

An example is shown in Figure 5, where the language L = {aba} in Figure 5a is realized by the minimal realization automaton in Figure 5b, then mapped into the network sketched

in Figure 5c, and finally into the coupled model in Figure 5d. There are two occurrences of symbol a, viz, a_1 and a_2, one occurrence of b, and states s0, . . . , s3 with s0 being the initial, and s3 the final state. In the network realization, there are two EPU recognizers for arrival of a symbol and one EPU for b and there are activation nodes for each of the four states. The external input is sent to each of the EPUs and when input arrives, there is exactly one EPU that is in the active state which recognizes it and passes on the activation to successor states via the corresponding activation nodes.

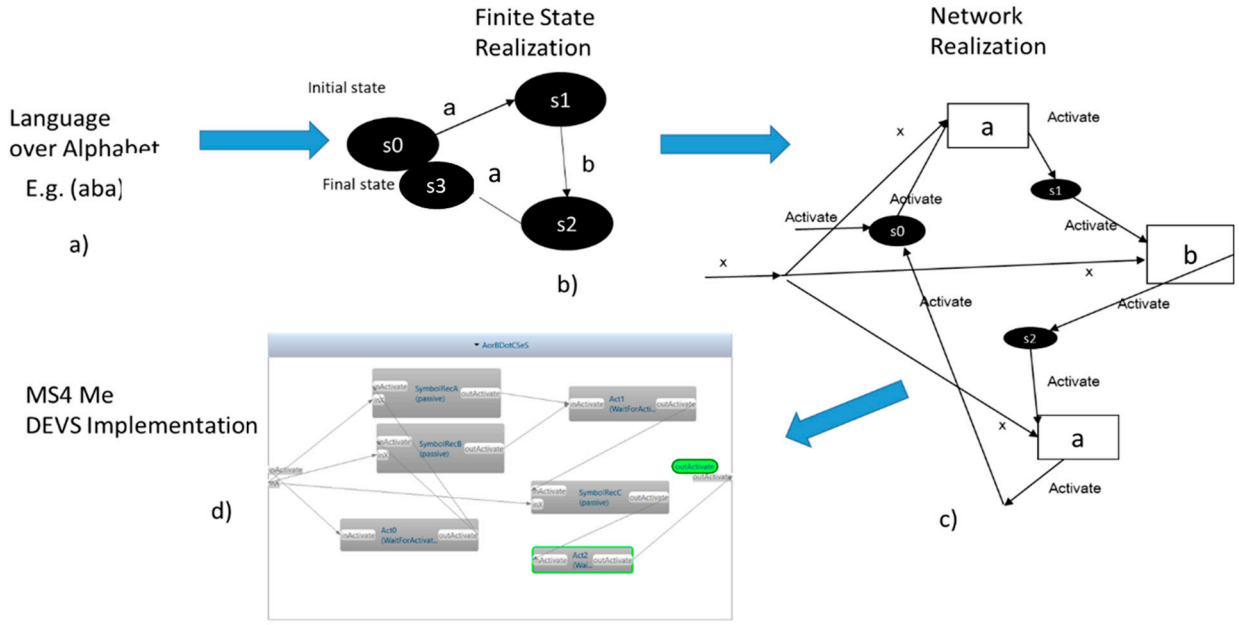

Figure 5. Mapping chain from FSA to DEVS implementation mediated by MetaSES. The automaton in (**a**) is realized by the minimal realization automaton in (**b**), then mapped into the network sketched in (**c**), and finally into the coupled model in (**d**).

3.2. Resolving Competing Solutions in Language Acceptance

The phenomenon of conflicting solutions, ubiquitously occurring in decision making models, actually shows up in the implementation of FSAs discussed thus far. Resolving such conflicts in the context of the DEVS realization will show how the *behavior level of the system specification hierarchy impacts the MetaSES at the high level*. For example, were the final state in Figure 5 to be incident with the initial state the accepted language would become L = {aba} *. In the usual formulation, a string such as abaaba would be accepted rather than the acceptable substring aba. This is because the end of the string is assumed to be detectable to the acceptor and it stops only when the end is detected. However, in the case of discrete event segments, an end marker is not available unless explicitly provided as an input event. In the following we extend the development of [5] to show how to resolve this problem.

To start, we review the formulation of language acceptance starting at the I/O Function level of the hierarchy of systems specifications (Figure 6a). In Figure 6b we display some I/O segment pairs and a simple automaton example in Figure 6c. In the latter, the initial state is s0 and the final states are s1 and s2 so that the accepted language is L = {a, aa}. The I/O Function that implements language acceptance outputs an Activate sometime after the end of the input segment. As illustrated in the first I/O pair of 6b, we use the absence in further input within a set duration (shown as $t_2 - t_1$) to make this judgement. As shown in in the second I/O pair in Figure 6b, the occurrence of an event within this interval (here a second letter a) should suppress the output that would have occurred, thus enabling continuation of processing, in this case to accept the segment representing successive letter a's.

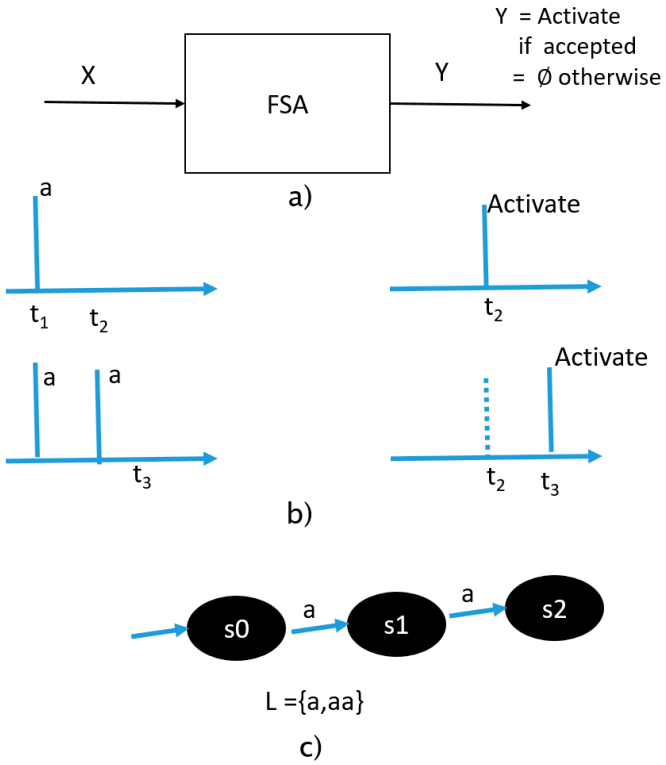

Figure 6. Illustration of a language requiring detection of segment end. The I/O Function (**a**), some I/O segment pairs (**b**) and automaton (**c**).

To implement this approach in a manner that is compatible with the generic formulation of the MetaSES, we extend the Activation node as illustrated in Figure 7, calling it Activation Node with Hold (ANWH). As before, it can send an Activate signal after being activated and as before, this signal goes to the EPUs dictated by the transitions out of the state it represents in the FSA. However, after a duration, HoldTime, it is also scheduled to send a Spike output–unless de-activated by a DeActivate input. In a composition, the Spike gets released downstream to indicate that the input segment has been recognized. This will happen unless an external input event arrives and deactivates the node thus recognizing that the segment is not yet over.

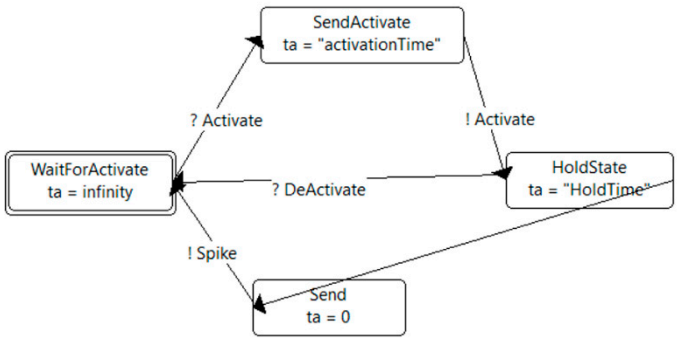

Figure 7. Implementation at state level of requirements of Figure 6.

The MetaSES is then extended to reflect the impact of the extended class as shown in Figure 8. We see additional couplings that enable the external input also to (1) send a De-Activate signal to ActivateNodeWHold elements (later these will be restricted to those representing final states) and (2) to send an Activate signal to the First Arrival node that is different from the Activate signal that is sent to the appropriate EPUs.

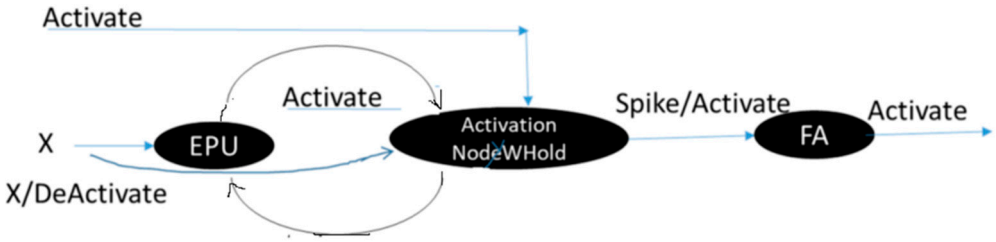

Figure 8. Extended MetaSES to match requirements of Figure 7.

Officially, the extensions to the MetaSES and the Transformable SES are given here:

MetaSES: Given FSA = <Q,X,delta,q_o,F>

- Coupling:
  - FSA sends X as DeActivate to AN
  - AN sends Spike as Activate to FA

Transformable SES:

- Coupling:
  - For each q in F, FSA sends X as DeActivate to AN_q
  - For each q in F, AN_q sends Spike as Activate to FA,
  - FA sends Activate to FSA

Note that although we have not shown this to be true, the solution is a minimal realization of the required behavior and therefore must be homomorphically represented in any realization- the minimal realization of the DEVS version requires |F| more states. We see that contrary to current neuromorphic representations the architectural coupling patterns characterized in the MetaSES must enable stimuli both to serve as sensory inputs as well as deactivation events and activations to be sent from the same source at different times from different output ports to different receivers.

Our approach has illustrated how it is necessary to span the levels of the specification hierarchy to simultaneously take account of both structure and behavior needed to achieve desired functionality. This includes consideration of how the solution is reflected at the level of the MetaSES.

3.3. Finite State Language Generation

As shown in Figure 9, by replacing perception by generation, we obtain Elementary Generation Units (EGU)s that produce output symbols when activated and interact with Activation Nodes in the same manner as for language recognition. As shown below, the MetaSES in this case is essentially the same as for the latter except for the output of symbols rather than activation.

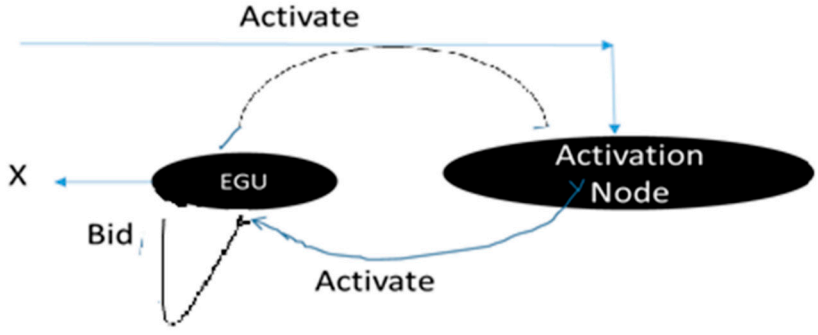

Figure 9. MetaSES for language generation.

MetaSES: Given FGA = < Q,X,delta,q_o,F>

- Classes: AN (Activation Node), Elementary Generation Unit (EPU)
- Coupling:
  - FGA send Activate to AN (external input coupling)
  - EGU sends X to FGA (external output coupling)
  - EGU sends Activate to AN,
  - AN sends Activate to EGU,
  - EGU sends Bid to EGU

The Free Digraph pre-image and the transformable SES follow the same pattern as for the language recognition case. The additional interaction involving the Bid message is discussed next.

3.4. Resolving Competing Outputs in Language Generation

The Elementary Generation Unit (EGU) shown in Figure 10, outputs its designated symbol after being activated. Competition arises when two or more EGUs are activated simultaneously as happens when they represent transitions emerging from the same state in the underlying FGA. To resolve this competition, EGUs first bid for the right to send their output with the winner determined as the one that first emits the bid output with the time randomly determined. This accounts for the self-coupling for the Bid message shown in the MetaSES of Figure 10.

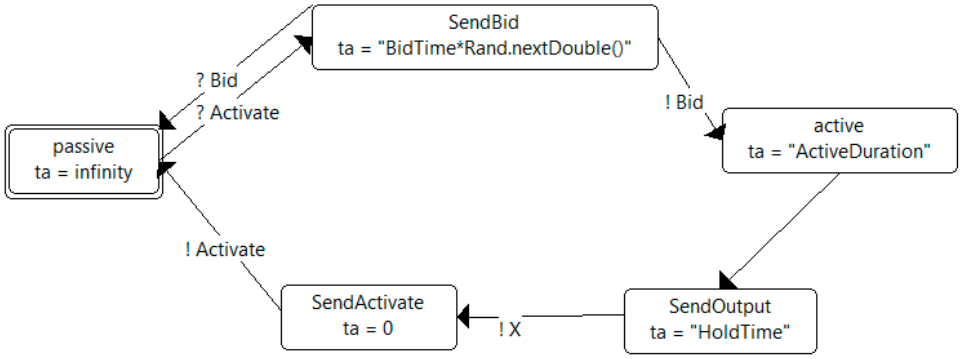

Figure 10. Implementation at state level of requirements of Figure 10.

Appendix A.2 provides the SES and DEVS Network Model generated for recognition of L = {a,aa} as an example of generating the transformable SES and from an initial MetaSES:

3.5. Summary: MetaSES Enhancement of System Specification

In summary, we have shown how the concept of MetaSES provides a high-level abstraction that enhances the hierarchy of system specifications as illustrated in Figure 1. The particular example of language acceptance and generation has illustrated how the concept enables abstract representation of the common structural features of a wide family of models (regular language devices) as well as considerations of the necessary modifications of the common basis to achieve desired functionality (specific recognition and generation). Further, as illustrated in Appendices A.1 and A.2, tools can support the transformation of MetaSES specifications through the identified stages into executable simulation models. Many examples, including for the regular language case, have been implemented as a proof-of-concept in MS4 Me [15]. In the sequel, we consider further extensions of the concept to hierarchical compositions and then as a basis for further research.

4. MetaSES for Hierarchical Compositions

Since the SES supports hierarchical construction and the MetaSES specifies SESs, it can also specify hierarchical compositions. We will formulate a MetaSES construction for a hierarchically constructed neural net model that is closer to real biological counterparts than typical artificial models to show the benefits of such construction. In [16] the neuron

model is composed of *dendrites*, *cellbody* and *axons*. *Dendrites* receive input messages from other neuron's axons and transmit them to the cellbody which releases a fire signal to its attached axons when threshold-type conditions are met. Using DEVS's dynamic structure capability, the cellbody can alter the axon structural composition in learning mode. Axons produce output weighted with respect to the available neurotransmitter.

As illustrated in Figure 11, the MetaSES for both network and neuron levels the extensions to the Transformable SESs are given here:

MetaSES for NN

- *Classes*: Neuron, N
- *Coupling*:
  - Neuron sends Spike to Neuron

MetaSES for Neuron

- *Classes*: Dendrite (D), CellBody (CB), Axon (A)
- *Coupling*:
  - N sends Spike to D, (external input coupling)
  - D sends Spike to CB,
  - CB sends Fire to A,
  - A sends Spike to N, (external output coupling)

Transformable SES for NN:

- *Components*:
  - Neurons, N
- *Coupling*:
  - All Neuron sends Spike to all Neuron

Transformable SES for Neuron, N

- *Classes*: Dendrite (D), CellBody (CB), Axon (A)
- *Coupling*:
  - N sends Spike to all D, (external input coupling)
  - All D sends Spike to CB,
  - CB sends Fire to all A,
  - All A sends Spike to N, (external output coupling)

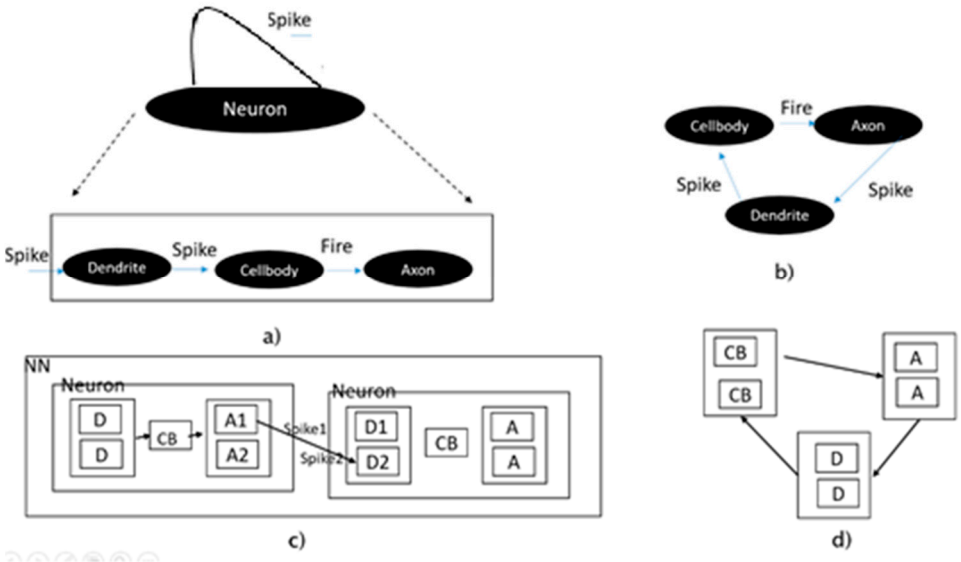

Figure 11. MetaSES for both network and neuron levels and illustration of axiom to dendrite coupling. (**a**) MetaSES for Neural Net, (**b**) Neuron, (**c**) Neural Net, (**d**) Resultant Coupled Model.

In the implementation of (Vahie 2000) each dendrite receives its inputs from some other neuron's axonal output connections. Further, each dendrite can handle up to three input connections and each connection has its own weighting factor. To represent this in the transformable SES, we expose the axon output ports to become visible at the neuron level.

Instead of "All A sends Spike to N" we have

- For each a_i in components of A, a_i sends Spike to N as $Spike_i$.

Similarly, instead of "N sends Spike to all D", we have

- For each d_i in components of D, N sends $Spike_i$ to d_i.

Then, we can limit couplings of dendrites of neuron N to axons of neuron N' to any specified subset of couplings from identified output ports of N to identified input ports of N'.

5. MetaSES and Model Continuity: DEVS Random Forest AI/ML Models

Random Forest models are collections of binary decision trees that classify input data sets, showing superior performance with data sets of very high variance [17]. (A binary decision tree is a digraph G = <Nodes,Edges> where for any edge we have eith((er outdegree(edge) = 2 (for interi-or nodes) or outdegree(edge) = 0 (for leaves.) Each node has a non-negative level (node) such that level(root) =0 and level(node) = length of path from root to the node.) Taking the trees as components, a forest becomes a *hierarchical parallel composition* of such trees. The MetaSES in Figure 12a helps us understand the similarities and differences between the tree component models and the network models just discussed. As in the case of language acceptance, the external input coupling has both Activate and sensory input, X in the MetaSES for binary decision tree models. As before, the Transformable SES consistent with the MetaSES provides the information needed to generate a simulateable DEVS model. This structure is built on the underlying decision tree with the Activate message connecting the root to the leaves by traversing the tree incrementally. As illustrated in Figure 12b, the external input X, is decomposed into features which are coupled to nodes with same label. Interior nodes are populated by YesNoDecision models [2] that compare their associated feature values against their assigned thresholds to determine the child node (left or right) to be activated. Channeled by such decisions, activation traverses a path from root to a leaf node, which reports the estimated class to which the input image belongs. Figure 12c illustrates the DEVS representation of a Random Forest as a parallel composition of decision trees which outputs to a collector that computes the winning class as the one getting the most votes. As simulation models, such compositions can be studied for their temporal behavior shown as suggested in Figure 12c (bottom).

The MetaSES approach facilitates executing the "model continuity" property of DEVS where the same specification employed for directing simulation model construction can be adopted for actual implementation as well. In the application underlying Figure 12d, we have implemented a workflow in which model structures (binary trees parameterized by thresholds and feature associations) are combined within the overall MetaSES pattern (Figure 12d) to create an SES that is transformed into a DEVS simulation model. On the other hand, the same information is used to write a Verilog specification of a Field-Programmable Gate Array (FPGA) implementation of the Random Forrest data classifier. Similar to the SES realization, the Verilog path "wires" components at the lower level of the system specification (interior node decision models and leaf classifier reporters) together to create the multi-level circuit realization. However, a notable difference in such "wiring" is that the SES manifests explicit coupling of input and output ports of modular DEVS components. In contrast, the Verilog equivalent of the SES is not so explicit as such ports are represented as parameters whose values are to be set by the appropriate parent–child connections required to form the binary tree. Referring to the hierarchy of system specifications of Figure 1, whereas the SES-based implementation exists at the Network of Systems Level (as a composition of modular systems), the Verilog implementation resides at the Multicomponent System Level where components are more tightly "wired" with

less ability to stand alone. We remark on the relative benefits of a truly modular DEVS hardware design process in the Discussion/Conclusions.

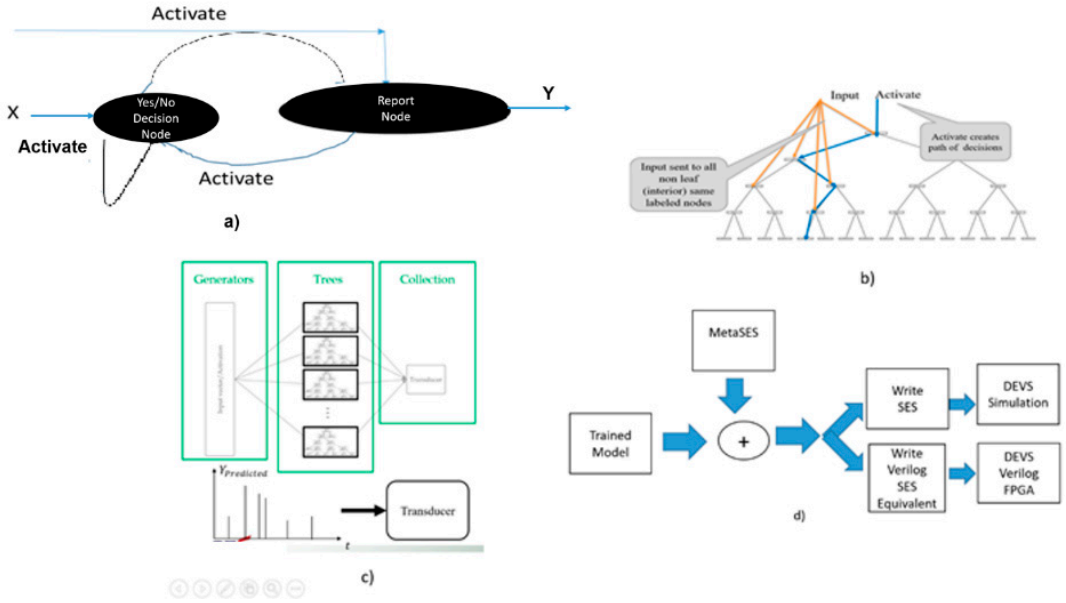

Figure 12. A MetaSES for Random Forest machine learning models in simulatable and hardware implementation form. (**a**) MetaSES for Random Forest, (**b**) Random Forest, (**c**) DEVS coupled model and behavior, (**d**) Workflow to generate final product.

6. Implication of MetaSES for Modeling and Simulation Methodology

Having elucidated the concept of MetaSES and its relation to the hierarchy of systems and morphisms, we turn to examining implications of this concept for aspects of modeling and simulation methodology including multi-resolution modeling, model repositories, cyber-physical system design and automated code generation. Research directions stemming from the proposed concepts are suggested in these contexts.

6.1. Implications for Multi-Resolution Modeling

Multiresolution modeling (MRM) helps to construct a collection of partial models, each oriented to one or more objectives as was illustrated with an example of Unmanned Autonomous Vehicles (UAV)s in a fleet possessing economic "earn your keep" mechanisms and modification of fleet size [18]. Higher-resolution models are created by removing assumptions that were previously added while including more refined representations to address the affected constraints and requirements. Meanwhile, checks for consistency of predictions between related base and lumped models are made using appropriate morphisms. The targeted base model is the one that would be achieved if, and, when all simplifying assumptions that have been made are removed in this iterative process. However, in a dynamic environment, the emerging family of lumped models provides the flexibility needed to address the various requirements as they unfold in lieu of an idealized base model. The hierarchy of system specifications and morphisms/SES provides the mathematical and computational concepts and tools needed for MRM methodology. The MetaSES proposed here enhances the hierarchy/SES to enable the construction of more complex lumped models and their integration in the family of models for a given SoS. Research on the MRM methodology and the MetaSES is needed to work out the concept and tool support needed to bring these concepts into the practice of modeling and simulation.

6.2. Implications for Simulation Modeling Repositories

Model repositories that ease the process of storing and reusing previously developed artifacts can usher in a new paradigm in model development with numerous advantages [19]. As mentioned above, repositories based on meta-model representations of the various ingredients of simulation experiments improve knowledge sharing across application domains and simulation approaches [9]. However, challenges persist in developing such repositories for modeling artifacts in general [20] and simulation models in particular [19]. In the meantime, a large backlog of legacy simulation models has accumulated over the years that is potentially exploitable for re-use in solving newly arising problems at relatively low cost and quick turn-around. However, the interpretation of model applicability and integration of models with others in a repository is a labor-intensive activity currently performed only by human modelers with assistance of subject matter experts. On the foundation of the hierarchy of system specifications and morphisms, the relation between simulation models and *experimental frames* (EF) has been formalized to support frame applicability to models in the service of composability and reuse. This provides the basis for a methodology to semantically characterize simulation models to support discovery and composition for legacy model curation [2]. Further, [21] presented a computational approach to represent and evaluate approximate parameter morphisms between pairs of models in support for such operations. Further research on data structures based on MetaSES concepts is needed to enable creating base/lumped pairs and their imposition of an organization of models in a repository.

6.3. Implications for Cyber-Physical System Design

The Discrete Event System Specification (DEVS) formalism has been recognized to support generic open architectures that allow incorporating multiple engineering domains within integrated simulation models. [22]) developed a modeling framework based on Model-Based System Engineering [23] methodology providing a clear separation between the specification of the models and their (computational) implementation. This improves the understanding of the system behavior and favors the incremental design and refinement of models facilitating scalability, maintainability, and reusability. The objective of this development is to overcome the limitations of non-DEVS optimizers that reduce the space of possible solutions to dynamic optimization problems. An example, optimizing UAV trajectories and sensor strategies in target-search missions require exploring large numbers of alternatives and large scale environments efficiently. While demonstrating the power of DEVS to support such exploration, Bordón-Ruiz et al. [24] in concert with many other DEVS-based simulation developments, do not take the next step of deploying the SES to accelerate the exploration. Exceptions exist such as Folkerts et al. [11]. The composability reusability feature of SES development methodology results in significant reduction in time to develop families of models to explore complex design spaces [12]. An essential feature of the SES in supporting exploration of models with large numbers of heterogeneous components is the concept of *multi-aspects*. This concept serves to describe the decomposition of an entity into specifiable numbers entities of the same class (see [10] for complete exposition). The MetaSES introduced here expands on this concept to enable a higher level of specification that maps into multi-aspects at the next lower level. Additionally, since the end results of MetaSESs are ordinary SESs the latter can be employed within more inclusive SESs [15]. This was illustrated by the parallel composition of decision trees in Figure 12.

6.4. Implications for Automated Code Generation

We have mentioned above how tools can be developed to further support the development of the MetaSES concept and its integration into the hierarchy of models and morphism for more automated development. Such development should follow practices of metamodeling, model transformation, and code generation, as, e.g., demonstrated by Alshareef et al. [25] in their implementation of a capability to create activity diagrams and to map them into executable activity-based models in the DEVS Markov formalism. Software

environments such as Eclipse and its modeling framework host plugins that enable model development based on metamodel descriptions [26,27] and the generation of executable code from such descriptions [28,29]. These linguistic tools can be applied to the elaboration of the MetaSES through the stages described above and illustrated in the Appendix A. Implementations of this kind can extend existing implementations of the SES such as in Java, [15], Python [11] or XLanguage [30]. However, experience has shown that there are limitations in such tools that prevent translations that are possible by directly coded string manipulation. Questions about how to manage such transformations need to be addressed through direct experience.

7. Conclusions

We have developed a new concept, that of MetaSES, that aims to achieve a unification of the hierarchy of system specifications and morphisms and the System Entity Structure (SES). The MetaSES supports the construction of complex systems of systems with multiple components belonging to specified classes. We demonstrated computational tools that need to be developed toward the goal of making it easier to design and build complex hierarchical DEVS models and to communicate their structures and intended behaviors to foster continued reuse and development. Several examples of application were given to illustrate how the MetaSES-based enhancement of the hierarchy of system specifications and morphisms helps to push the boundaries of complexity management in theory and practice of modeling and simulation toward the enhanced capability to explore large design spaces. Such design spaces are characteristic of problems with deep uncertainty with inadequate or incomplete information about the system and the outcomes of interest [31]. Formulation of such problems employing the MetaSES and the hierarchy of systems specifications and morphisms is proposed here to help conduct exploratory modeling and simulation analysis in the search for robust solutions addressing new challenges in the theory and practice of modeling and simulation.

Application to the exploration of large design search spaces will drive the need to research and develop tools to support MetaSES concepts as they evolve. In our discussion of the implication of MetaSES for modeling and simulation methodology, we considered some areas of application and suggested open research directions. We indicated that research on the MetaSES is needed to work out the concept and tool support needed to bring these concepts into wider use of multi-resolution modeling methodology. In discussing implications for simulation modeling repositories, we indicated that further research on data structures based on MetaSES concepts is needed to enable the creation of base/lumped pairs and their imposition of an organization of models in a repository. For implications in the cyber-physical system design area, we noted that the MetaSES enables a higher level of system specification that maps into multi-aspects at the next lower level and that more research is needed on how to support and exploit such specifications. Finally, research is needed to apply automated code generation to elaborate the MetaSES through the designated stages. However, we noted that although developments of this kind can extend existing implementations of the SES, there appear to be limitations in such tools and that questions about how to manage such transformations need to be addressed through direct experience in their application.

Funding: This research was partially supported by RTSync internal funding and received no external funding.

Data Availability Statement: Data sharing not applicable.

Acknowledgments: I would like to thank Sang Won Yoon, and Christian Koertje, Graduate Research Associate at the Watson Institute for Systems Excellence at The State University of New York at Binghamton, NY, and Tosiron Adegbija, and Kama Laruen Svaboda of the Electrical and Computer Engineering Dept., the University of Arizona, for their help in working on the Random Forest example.

Conflicts of Interest: The author declares no conflict of interest.

Appendix A.

Appendix A.1. Transformation of MetaSES Specifications through the Identified Stages into Executable Simulation Models

Construction of Coupled DEVS model from MetaSES description

We construct the free digraph represented by the MetaSES and then the coupled model derived from it. For simplicity we consider only internal couplings. The mapping to external couplings can be handled with the same approach.

MetaSES G = <Nodes,Edges> where Nodes = Classes = {A}, e ∈ Edges implies e = (A,(op,ip),B), and A, B ∈ Classes (where we allow A = B).

FreeDigraph generated by G: Given cardinality of A, $|A|$, for each A ∈ Classes, we have FG = < Nodes,Edges> where Nodes = Disjoint Union of Components of A in Classes, where components of A = $\{a_1, a_2, \ldots, a_{|A|},\}$ and Edges = { $(a_i,(op,ip), b_j) | a_i, \in$ Components of A, $b_j, \in$ Components of B, and e = (A,(op,ip),B) is an edge in G.

The Coupled DEVS Model constructed from FG,

N(FG) = < D, $\{M_d | d \in D$, IC>

Where D = Nodes of FG and for each d ∈ D, M_d is a DEVS model with input ports, IP_d, and output ports, OP_d.

Here, IP_d = {ip | there is an edge ∈ G of the form (A, (op,ip), B), where d ∈ B } (ip is the input port for some specified coupling)

OP_d = {op | there is an edge ∈ G of the form (A, (op,ip), B), where d ∈ A } } (op is the output port for some specified coupling)

For each edge e = (A,(op,ip),B) ∈ Edges of G, each component a_i of A and each component b_j of B, we add a coupling pair $(a_i,(op,ip), b_j)$ to IC. Thus we have all-to-all coupling from components of A to components of B for every pair of classes A, B mentioned in the MetaSES, G.

Thus, IC = < $((a_i,op), (b_j,ip)$ | $(a_i,(op,ip), b_j)$ is an edge of FG>

Algorithm sketch to write the Free SES i given MetaSES, G

Given DEVS models M_A, M_B, M_C,.. for Classes A,B,C, . . .
and given respective sizes N_A,N_B,N_C, . . . define an SES to construct the DEVS coupled model N(FG)
Define a string s = “”;

1. Create HashSets for each class, X ∈ Classes

HS_X = {id0_X, id1_X,..,}

2. Write the first line for the SES:

s + = “From the system perspective, N is made of “;
For each X ∈ Classes
For each x ∈ HS_X
s + = x+”,”;
s + = “!”;

3. Write the internal coupling:

For each pair (A,B) for which there is an edge e = (A,(op,ip),B) ∈ G
For each a ∈ HS_A
For each b ∈ HS_B
s + = “From the system perspective,
“+a+” sends “+op+” to “+b+” as “+ip+”!”

Appendix A.2. Example of SES and DEVS Network Model Generated from MetaSES: Recognition of L = {a,aa}

From the fsa perspective, FSAAARecNet is made of FirstArrival, XN1_EPU,XN2_EPU, passive_ActivationWHold, TwoA_ActivationWHold, and OneA_ActivationWHold!!
From the fsa perspective, passive_ActivationWHold sends Activate to XN1_EPU!

From the fsa perspective, XN1_EPU sends Activate to OneA_ActivationWHold!
From the fsa perspective, OneA_ActivationWHold sends Activate to XN2_EPU!
From the fsa perspective, XN2_EPU sends Activate to TwoA_ActivationWHold!
From the fsa perspective,FSAAARecNet sends Activate to passive_ActivationWHold!
From the fsa perspective, OneA_ActivationWHold sends outSpike to FirstArrival as inX!
From the fsa perspective, TwoA_ActivationWHold sends outSpike to FirstArrival as inX!
From the fsa perspective, FirstArrival sends outY to FSAAARecNet as outActivate!
From the fsa perspective, FSAAARecNet sends inX to XN1_EPU as inSpike!
From the fsa perspective, FSAAARecNet sends inX to OneA_ActivationWHold as inDeActivate!
From the fsa perspective, FSAAARecNet sends inX to TwoA_ActivationWHold as inDeActivate!
From the fsa perspective, FSAAARecNet sends inX to XN2_EPU as inSpike!
From the fsa perspective, FSAAARecNet sends inX to OneA_ActivationWHold as inDeActivate!
From the fsa perspective, FSAAARecNet sends inX to TwoA_ActivationWHold as inDeActivate
The network DEVS model generated from this SES is shown in Figure A1.

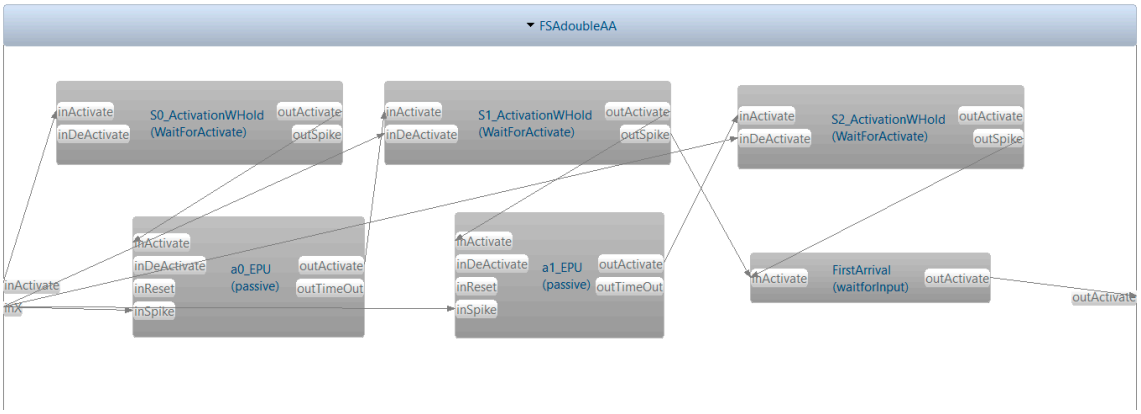

Figure A1. The network DEVS model generated from this SES.

References

1. Oren, T.; Zeigler, B.P.; Tolk, A. (Eds.) *Body of Knowledge for Modeling and Simulation*; Springer: Berlin/Heidelberg, Germany, 2022.
2. Zeigler, B.P. Exploiting the levels of system specification for modeling of mind. In Proceedings of the Winter Simulation Conference (WSC), Singapore, 11–14 December 2022.
3. Zeigler, B.P.; Muzy, A.; Kofman, E. *Theory of Modeling and Simulation: Discrete Event Iterative System Computational Foundations*; Academic Press: New York, NY, USA, 2018.
4. Wymore, W.A. *A Mathematical Theory of Systems Engineering—The Elements*; Wiley: Hoboken, NJ, USA, 1967.
5. Zeigler, B. DEVS-Based Building Blocks and Architectural Patterns for Intelligent Hybrid Cyberphysical System Design. *Information* **2021**, *12*, 531. [CrossRef]
6. Grossberg, S. *Conscious Mind/Resonant Brain: How Each Brain Makes a Mind*; Oxford University Press: New York, NY, USA, 2021.
7. Minsky, M. *The Society of Mind*; Simon and Schuster: New York, NY, ISA, 1986.
8. Wang, Y. On Cognitive Informatics. *Brain Mind* **2003**, *4*, 151–167. [CrossRef]
9. Wilsdorf, P.; Heller, J.; Budde, K.; Zimmermann, J.; Warnke, T.; Haubelt, C.; Timmermann, D.; van Rienen, U.; Uhrmacher, A.M. A Model-Driven Approach for Conducting Simulation Experiments. *Appl. Sci.* **2022**, *12*, 7977. [CrossRef]
10. Pawletta, T. Basic System Entity Structure (SES) Concepts. In *Body of Knowledge for Modeling and Simulation*; Oren, T., Zeigler, B.P., Tolk, A., Eds.; Springer: Berlin/Heidelberg, Germany, 2022; Chapter 1.5.
11. Folkerts, H.; Pawletta, T.; Deatcu, C.; Santucci, J.; Capocchi, L. An Integrated Modeling, Simulation and Experimentation Environment in Python Based on SES/MB and DEVS. In Proceedings of the SummerSim-SCSC, Berlin, Germany, 22–24 July 2019.
12. Keller, N.; Zeigler, B.; Kim, D.; Anderson, C.; Ceney, J. Supporting the reuse of algorithmic simulation models. In Proceedings of the Spring Simulation Conference, Baltimore, MD, USA, 15–18 April 2018.
13. Zeigler, B.P.; Sarjoughian, H. *Modeling and Simulation of Systems of Systems*; Springer Pub. Co.: New York, NY, USA, 2017.
14. Zeigler, B.P. Hybrid Iterative System Specification of Cyberphysical Systems: Neurocognitive Behavior Application. In Proceedings of the 2020 Spring Simulation Conference (SpringSim), Virtual Conference, Fairfax, VA, USA, 18–21 May 2020.
15. Seo, C.; Zeigler, B.; Coop, R.; Kim, D. DEVS Modeling and Simulation Methodology with MS4Me Software TMS. In Proceedings of the 2013 Spring Simulation Multiconference, San Diego, CA, USA, 7–10 April 2013.

16. Vahie, S. Dynamic Neuronal Ensembles: Neurobiologically Inspired Discrete Event Neural Networks. In *Discrete Event Modeling and Simulation Technologies*; Sarjoughian, H., Ed.; Springer: Berlin/Heidelberg, Germany, 2000.
17. Alam, M.S.; Vuong, S.T. Random Forest Classification for Detecting Android Malware. In Proceedings of the IEEE International Conference on Green Computing and Communications, Beijing, China, 20–23 August 2013; pp. 663–669.
18. Zeigler, B.P.; Kim, D. Multi-Resolution Modeling for Adaptive UAV Service Systems. In Proceedings of the 2019 Spring Simulation Conference (SpringSim), Tucson, AZ, USA, 29 April–2 May 2019; pp. 1–12. [CrossRef]
19. Valdemar, V. Graciano Neto and Cláudio Gomes, 2022 Resource Repositories. In *Body of Knowledge for Modeling and Simulation*; Oren, T., Zeigler, B.P., Tolk, A., Eds.; Springer: Berlin/Heidelberg, Germany, 2022.
20. Basciani, F.; Di Rocco, J.; Di Ruscio, D.; Pierantonio, A.; Ludovico, I. Model repositories: Will they become reality? In Proceedings of the International Workshop on Model-Driven Engineering on and for the Cloud (CloudMDE), Ottawa, ON, Canada, 29 September 2015.
21. Zeigler, B.P. Simulation-Based Evaluation of Morphisms for Model Library Organization. In *Model Engineering for Simulation*; Lin, Z., Bernard, P.Z., Yuanjun, L., Eds.; Academic Press: Cambridge, MA, USA, 2019.
22. Gourlis, G.; Kovacic, I. Energy efficient operation of industrial facilities: The role of the building in simulation-based optimization. *IOP Conf. Ser. Earth Environ. Sci.* **2020**, *410*, 012019. [CrossRef]
23. Zeigler, B.P.; Mittal, S.; Traore, M. MBSE With/Out Simulation: State of the Art and Way Forward. *Systems* **2018**, *6*, 40. [CrossRef]
24. Juan, B.-R.; Besada-Portas, E.; López-Orozco, J.A. Cloud DEVS-based computation of UAVs trajectories for search and rescue missions. *J. Simul.* **2022**, *16*, 572–588. [CrossRef]
25. Alshareef, A.; Seo, C.; Kim, A.; Zeigler, B.P. DEVS Markov Modeling and Simulation of Activity-Based Models for MBSE Application. In Proceedings of the 2021 Winter Simulation Conference, Phoenix, AZ, USA, 12–15 December 2021.
26. Xpand Code Generation Based on EMF Models. 2021. Available online: https://projects.eclipse.org/projects/modeling.m2t.xpand (accessed on 13 May 2021).
27. Xtext Language Engineering for Everyone. 2021. Available online: https://www.eclipse.org/Xtext/ (accessed on 13 May 2021).
28. Sirius. The Easiest Way to Get Your Own Modeling Tool. 2021. Available online: https://www.eclipse.org/sirius/ (accessed on 13 May 2021).
29. Acceleo. Generate Anything from Any EMF Model. 2021. Available online: https://www.eclipse.org/acceleo/ (accessed on 10 November 2022).
30. Xie, K.; Li, X.; Zhang, L.; Gu, P.; Chen, Z. SES-X: A MBSE methodology based on SES/MB and XLanguage. *Inf. J.* **2023**, *14*, 23.
31. Tolk, A. Simulation-Based Optimization: Implications of Complex Adaptive Systems and Deep Uncertainty. *Information* **2022**, *13*, 469. [CrossRef]