DEVS MARKOV MODELING AND SIMULATION: FORMAL DEFINITION AND IMPLEMENTATION

Chungman Seo
RTSync Corp.
6909 W Ray Rd STE 15-107
Chandler, AZ, USA
cseo@rtsync.com

Bernard P. Zeigler
RTSync Corp.
6909 W Ray Rd STE 15-107
Chandler, AZ, USA
zeigler@rtsync.com

Doohwan Kim
RTSync Corp.
6909 W Ray Rd STE 15-107
Chandler, AZ, USA
dhkim@rtsync.com

ABSTRACT

Markov Modeling is among the most commonly used forms of model expression and Markov concepts of states and state transitions are fully compatible with the DEVS characterization of discrete event systems. Besides their general usefulness, the Markov concepts of stochastic modeling are implicitly at the heart of most forms of discrete event simulation and are a natural basis for the extended and integrated Markov modeling facility discussed in this paper. DEVS Markov models are full-fledged DEVS models and can be coupled with other DEVS components in hierarchical compositions. Due to their explicit transition and time advance structure, DEVS Markov models can be individualized with specific transition probabilities and transition times/rates which can be changed during model execution for dynamic structural change. This paper presents the formal concepts underlying DEVS Markov models and how they are implemented in MS4 Me, also discussing how the facilities differ from other Markov M&S tools.

Keywords: Markov Modeling, MS4 Me, DEVS Markov Model, Stochastic Modeling, State Designer, DEVS Natural Language

1 INTRODUCTION

Markov Modeling is among the most commonly used forms of model expression and many classes have been studied, including Markov Chains (Feller 1966), Continuous Time Markov (CTM) Processes(Kemeny and Snell 1960), Discrete Time Markov (DTM), Semi-Markov Processes (Barbu and Limnios 2008), Generalized Semi-Markov Processes (Glynn 1989;Younes and Simmons 2004), Markov Decision Processes (Rachelson, E. et al. 2008;Puterman 1994), and Hidden Markov Models (Barbu and Limnios 2008). DEVS Markov Models, as a subclasss of Stochastic DEVS (Castro et al 2010), can represent complex systems at the level of individual subsystems and actors. In this guise, each system or actor can be represented as a component with states and transitions as well as inputs and outputs that enable it to interact as atomic models within coupled models using coupling in the usual way. Briefly stated, these atomic and coupled models are useful because:

- The Parallel DEVS (PDEVS) substrate supports coupled models with concurrent atomic model interaction.
- The PDEVS simulator provides a Monte Carlo layer that generates stochastic sample space behavior simulator.
- DEVS Markov models can express probabilistic agent-type alternative decisions and consequences, generate and analyze both transient and steady state behavior.
- Together with experimental frames, DEVS Markov models support a wide range of performance metrics (including the usual queueing related ones such as waiting times, throughput, losses.)

Aggregations of such models, state-transition matrix models, are computationally much faster because they employ deterministic computation of probabilities interpreted as frequencies of state occupation of the corresponding DEVS Markov models. Such models add to the modeling capability because:

- They yield probabilities for ergodic behaviors in steady state
- They yield probabilities models that reach absorbing states
- They support computation of state-to-state traversal times for models where time consumption is of essential interest
- They provide simplifications that are accurate for answering certain questions and can be composed to yield good approximations to compositions of DEVS Markov models.

To implement DEVS Markov modeling and simulation, MS4 Me (Seo et al. 2013; Zeigler et al. 2017) was upgraded to support designing DEVS Markov models using its State Designer with states and probabilistic transitions. DEVS Markov models further augment this version by graphically supporting specification of distributions functions that are required to randomly select transition times. For this purpose, Markov supporting packages were added to the MS4 Me environment. A Markov design can be generated using the state diagram which generates a DEVS Natural Language (DNL) file and an XML document containing state transition probability and time information for each transition. MS4 Me also automatically generates a state-transition Matrix Model (MM) from the XML document.

In the rest of the paper, section 2 characterizes the variety of DEVS Markov Models using a System Entities Structure (SES). Section 3 illustrates the formalization of DEVS Markov models with a simple example. Details of the MS4 Me environment and implementation of DEVS Markov models appear in section 4. Discussion of DEVS Markov models in relation to other implementations are in the section 5 with conclusions in section 6.

2 DEVS MARKOV MODELS

Integrating Markov modeling into DEVS opens up a wide variety of model types that can be incorporated within the same framework. It helps to organize such models into classes that relate both to the traditional ones encountered in the mathematics and applications literature as well as to the structural features that characterize all DEVS models (Zeigler, Muzy, and Kofman 2018; Hwang and Zeigler 2009) as specifications of input/output dynamic systems. In the following, we employ a System Entity Structure (Zeigler and Hammonds 2007;Zeigler et al. 2013) to provide such an organization. It sees a DEVS Markov model specification as composed of a time base, phase set, external event set and transition function mapping with specializations for each component. Classes and sub-classes of such models then can be formed by choice of elements within some of specializations, perhaps leaving others unspecified. The broader the class, the fewer elements of the SES are fixed. Conversely, the narrower the class, the greater are the number of selections made.

2.1 SES for DEVS Markov Models

The specializations in our SES for DEVS Markov models shown in Figure 1 are;

- ***Time Base*** – can be discrete or continuous. Most of our concentration will be on continuous time but simplifying properties of discrete time make it useful at times.
- ***Phase Set*** – this is the state set typically referred to in math expositions. The reason we refer to it as the phase set is that the state of a DEVS will include sigma, the explicit representation of the time advance. In addition, the global state of DEVS includes the elapsed time. We focus mostly on finite phase (Markov state) sets, however, much of the approach will extend to infinite state sets.
- ***External Event Set*** – this is the external interface including the input and output sets. Sometimes, it is convenient to consider only the transition portion of the DEVS structure, which omits the external event set related elements, and will be called the *core* of the structure. On the other hand, the external interface is needed for including model types such as hidden Markov and Markov Decision Processes that interact with the environment and other agents.

Also not shown

- ***External Transition Function*** – models can disallow the elapsed time since the last event to influence the effect of an input. This restriction is in effect for DTM and CTM as will be shown.
- ***Transition Function Mapping*** – this will help us to understand the differences in the varieties of Markov models in the literature, e.g., semi-Markov, GSMP(Nielsen 1998), hidden Markov, etc.

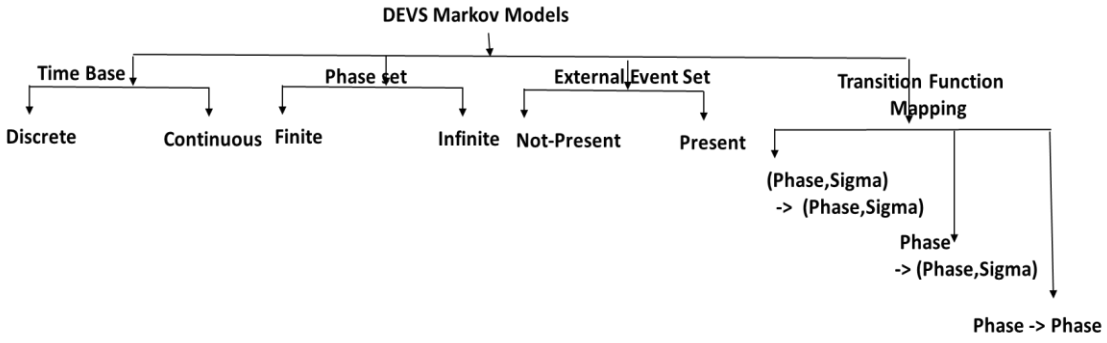

Figure 1: System Entity Structure for DEVS Markov Models

The kernel of the Markov implementation in DEVS is the Transition Function Mapping. At every internal transition the current phase and sigma together with the transition probabilities or rates that characterize the model together a random number input select a next phase and sigma (which play the roles of Markov state and sojourn or transition time, respectively.)

The flow chart in Figure 2 illustrates the control flow for the internal transitions for an atomic DEVS Markov model based on a pair of Probability Transition Structure (PTS) and Time Transition Structure (TTS).

Subclasses of Markov models can be characterized by pruning operations on the overall SES. The DEVS Semi-Markov model employs the most flexible transition mechanism which uses the current phase and sigma to select the next phase and sigma. When composed into coupled models such a models include the Generalized Sequential Markov Process (GSMP) (Glynn 1989; Rachelson et al. 2008). For example, the Discrete Time Markov (DTM) model class is generated where the time base is discrete with finite state set. The transition specification employs transitions from phases to phases which take a fixed time step also called a cycle length. In contrast the Continuous Time Markov (CTM) employs a continuous time base and a transition specification that employs only the current phase to determine both the next phase and the

transition time (Figure 3). It can so because the transition probabilities are interpreted as rates in exponential distributions for transition times. That is all transition times are based on the same type of distribution (exponential) while in the more general case, a time distribution can be associated with each non-self-transition pair (phase, phase').

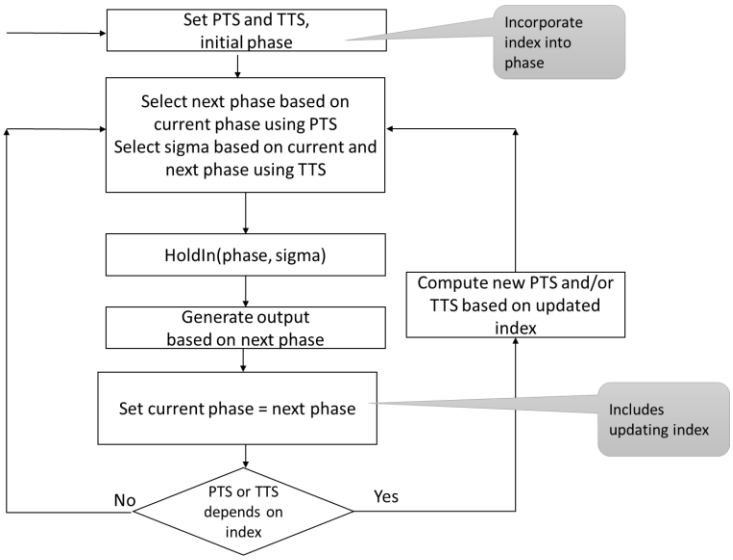

Figure 2: Flow chart to illustrate DEVS Markov atomic model

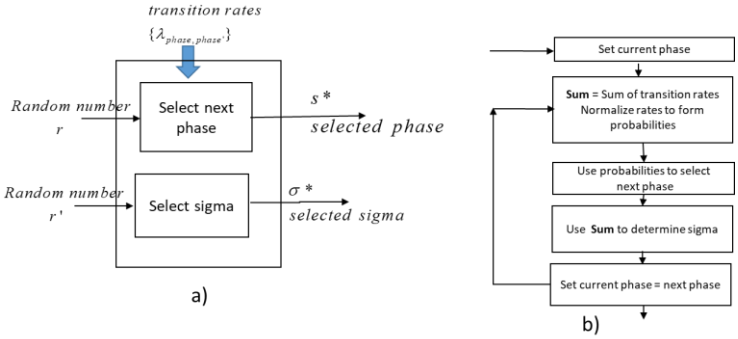

Figure 3: Selection of next phase and sigma for CTM

3 FORMALIZATION OF DEVS MARKOV MODELS

3.1 Probability Core DEVS

We introduce two probabilistic structures that will provide the basis for specifying the DEVS Markov formalism that will formalize the Transition Function Mapping of Figure 1. First, we define a *Probability Transition Structure (PTS)* in set-theoretic form to be given in a moment. It often takes on the familiar form of a matrix of probability values. For example, the matrix $\begin{bmatrix} p_{00} & p_{01} \\ p_{10} & p_{11} \end{bmatrix}$ is represented by the structure: $< \{0,1\}, Pr >$ where $Pr: \{0,1\} \times \{0,1\} \rightarrow [0,1]$ such that $\Pr(i,j) = p_{ij}$. Formally, a *Probability Transition Structure* is a structure $PTS = < S, Pr >$ where $Pr: S \times S \rightarrow [0,1]$ and $P\,r(s,s') = v, 0 \leq v \leq 1$. As a relation Pr contains triples of the form (s,s',v) which stands for state s transitions to state s' with probability v. For each $s \in S$, define the restriction of Pr to s, $\Pr \mid s : S \rightarrow [0,1]$ defined by $\Pr \mid s\,(s') =$

$Pr(s, s')$. Then *Pr* is subject to the constraint that it is fully defined (every transition has a probability) and the probabilities of transitions out of every state sum to 1. That is, for each $s \in S$,

$$\sum_{s' \in S} Pr\,|s(s') = 1$$

The set-theoretic representation of the usual matrix supports manipulations equivalent to data structure operations that render it more convenient to create and manage derived DEVS models.

3.2 Markov Chain

The basic interpretation of a PTS is of a Markov chain, i.e., a set of states that generate sequences determined by the probability structure. The probabilities of the ensemble of all such state sequences can be described by a vector representing probabilities of being in the states and iterative application of the associated matrix. In a very abbreviated summary of Markov chain theory (Dayar 2013; Banisch 2015), we have the state vector in equilibrium reached at step n^* is defined by $p(n+1) = p(n), for\ all\ n > n^*$ which implies that $p(n^*) \cdot P = p(n^*)$ and $p^* = p(n^*)$ is the equilibrium vector where a unique solution exists.

The second structure to be introduced allows us to work with the times of the transitions. These can be referred to variously and equivalently as sojourn times, transition times, time advances, elapsed times, or residence times depending on the context.

Time Transition Structure : $TTS =< S, \tau >$ where $\tau: S \times S \rightarrow ProbabilityDensityFunctions$ such that the time for transition from *s* to *s'* is selected from $\tau(s, s'): R^+_{0,\infty} \rightarrow [0,1]$.

$\begin{bmatrix} \tau_{00} & \tau_{01} \\ \tau_{10} & \tau_{11} \end{bmatrix}$ is represented by the structure:<{0,1}, τ> where $\tau: \{0,1\} \times \{0,1\} \rightarrow [0,1]$ such that $\tau(i,j) = \tau_{ij}$ and τ_{ij} is a *pdf* (probability distribution function) $\tau: R^+_{0,\infty} \rightarrow [0,1]$. For example, $\tau_{ij}(t) = e^{-t}$ represents the exponential pdf for selecting a time for transition from *i* to *j*.

The pair *(PTS, TTS)* specifies a DEVS Markov core as follows:

Probability Transition Structure : *PTS* = <*S*, *Pr*> and Time Transition Structure : *TTS* = <*S*, τ> gives rise to a DEVS Markov Core $M_{DEVS} =< S_{DEVS}, \delta_{int}, ta >$ where $S_{DEVS} = S \times [0,1]^S \times [0,1]^S$ with typical element (s, γ_1, γ_2) with $\gamma_1: S \rightarrow [0,1], i = 1,2$ where $\delta_{int}: S_{DEVS} \rightarrow S_{DEVS}$ is given by: $s' = \delta_{int}(s, \gamma_1, \gamma_2) = (SelectPhase_{PTS}(s, \gamma_1), \gamma_1', \gamma_2')$ and $ta: S_{DEVS} \rightarrow R^+_{0,\infty}$ is given by : $ta(s, \gamma_1, \gamma_2) = SelectSigma_{TTS}(s', \gamma_2)$ where note that *s'* is a function of *s*, and $\gamma_i' = \mathbb{\Gamma}(\gamma_i), i = 1, 2.$

3.3 Example of DEVS Markov Core

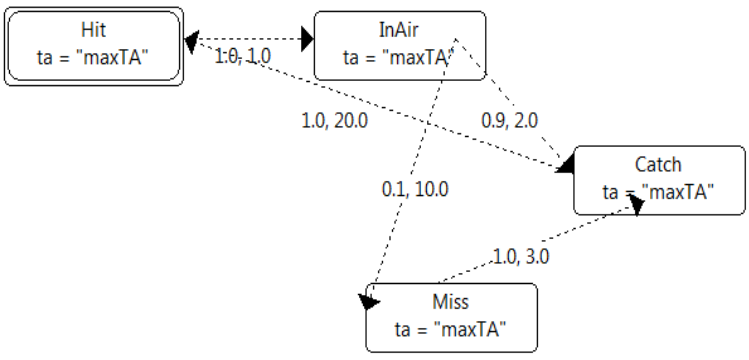

Figure 4: Markov Model Example: Baseball scenario

Consider the situation depicted in Figure 4 in which a fielder has a probability of catching or fumbling a ball. In contrast to the CTM approach depicted in the figure, using DEVS Markov formulation, we can separately account for the probability of one or the other eventuality as well as the times taken in each case. Let the phases be coded by integers as follows : $Hit = 0, InAir = 1, Catch = 2, Miss = 3$. Then the PTS and TTS can be represented by matrices :

$$\begin{bmatrix} p_{00} & p_{01} p_{02} & p_{03} \\ p_{10} & p_{11} p_{12} & p_{13} \\ p_{20} & p_{21} p_{22} & p_{23} \\ p_{30} & p_{31} p_{32} & p_{33} \end{bmatrix} \quad \begin{bmatrix} \tau_{00} & \tau_{01} \tau_{02} & \tau_{03} \\ \tau_{10} & \tau_{11} \tau_{12} & \tau_{13} \\ \tau_{20} & \tau_{21} \tau_{22} & \tau_{23} \\ \tau_{30} & \tau_{31} \tau_{32} & \tau_{33} \end{bmatrix}$$

The non-zero probability elements are : $p_{01} = 1, p_{12} = 0.1$, $p_{13} = 0.9$, $p_{20} = 1$, $p_{32} = 1$. Note that a transition that is the only outgoing one from a phase gets a probability of 1. Now let the elements of the TTS be distributions for transition times with mean values as follows : $\tau_{01} mean = 1, \tau_{12} mean = 2$, $\tau_{13} mean = 10$, $\tau_{20} mean = 20$, $\tau_{32} mean = 3$. With time units as seconds, these values assert that the ball is in the air for one sec., and if it is caught it takes 10 sec. to recover and be in a position to return it to the catcher. Using exponential *pdf*s we need only specify these values as parameters while other distributions might require specifying more parameters.

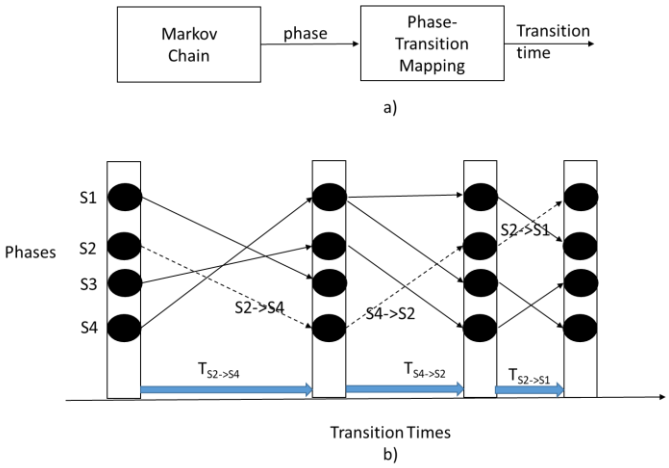

Figure 5: State trajectories for DEVS Markov core model and sequential decomposition

The conversion of the pair *(PTS,TTS)* into a DEVS Markov Core model is the formal equivalent of the flow chart in Figure 2. We see that to make a transition, first we randomly select a next phase based on PTS which specifies a transition pair. We then randomly select a sigma for the time of this transition. The random selections are performed by the functions *SelectPhase* and *SelectSigma*, respectively.

The sequential dependence of transition time selection dependence on probability selection and its evolution over time are illustrated in Figure 5 a) and b) respectively. A state trajectory can be visualized as starting in an initial phase such as S2, and advancing with variable time steps determined by the successive states. For example, the transition from S2 to S4 is selected based on transition probabilities out of S2 and then the transition time $T_{S2->S4}$ is selected from its distribution. This sequential dependence allows us to use the representation in Figure 5 a) where the successive selection of probabilities is considered as a Markov chain as front component with the time advance mapping as back component. Thus if we are only interested in the probabilities of states at transitions (and not in the timing of the transitions) then we can consult the Markov chain interpretation of the PTS, In particular for an ergodic chain, the steady state probabilities can be computed using the Markov Matrix model. Having these probabilities we can then compute the expected sojourn times using the TTS distributions.

3.4 Closure Under Coupling

It can be shown that the DEVS Markov class is closed under coupling. However, the smaller class DEVS CTM (and DTM to which it is equivalent) is not closed under coupling. Further, closure of the DEVS CTM class can be proved when dependence on elapsed time is not allowed. Closure under coupling provides assurance that the class under consideration is well-defined and enables checking for the correct functioning of feedback coupled models. Absence of closure is also informative as it begs for characterizing the smallest closed class that includes the class. Here it can be shown that the smallest such class is the DEVS Markov class itself.

4 DEVS MODELING AND SIMULATION ENVIRONMENT

4.1 Review of MS4 Me

A review of the MS4 Me environment will help to provide some essential background for the design and implementation of the DEVS Markov modeling and simulation. MS4 Me is an environment to design general systems as well as Systems of Systems (SoS) based on Discrete EVent System specification(DEVS) modeling and simulation theory (Zeigler, Muzy, and Kofman 2018; Hwang and Zeigler 2009). MS4 Me supports collaboration of domain experts and modelers in both top down and bottom up system construction. To do so, it provides tools such as the state diagram designer, sequence diagrammer, and System Entity Structure (SES) pruning GUI, to generate DEVS atomic and coupled models. Top down design can start with an SES which describes the overall system as a tree structure, which comprises entities (which refer to components of the system) with relations (such as decomposition, specialization, and multi-aspect). To generate a specific coupled model (one instance from the SES), a pruning process (tailoring the system) needs to be done with a script containing pruning information. After transformation, entities become either atomic models (leaf nodes) or coupled models (middle nodes) recreated from existing Java classes. For bottom up design, atomic models are first developed and then the whole system is constructed with an SES. In this process, modelers can use a customized text editor to express atomic models using the DEVS Natural Language (DNL), a restricted form of natural language. Alternatively, the state diagram designer supports graphical specification of atomic models. Eventually, the state diagram is automatically (and reversibly) converted to a DNL file and a Java atomic model class is compiled to execute in the MS4 Me execution environment. Of course, alternation between top down and bottom up processes and iteration of elements in the workflow are also encouraged.

4.1.1 DEVS Natural Language (DNL)

The DNL focuses on the description of the DEVS model structure but also includes program language specific information in a structured modular manner. 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 computer code to be inserted in specific locations within the Java class file, e.g., within the characteristic functions of the DEVS Java model. Thus the tag blocks allow a DNL file to include programming specific information so as to enjoy full computer language expressive power.

4.1.2 State Designer

MS4 Me supports several types of DEVS atomic models which can be constructed with a state diagram to help users visualize states and transitions comprising the model. A normal DNL file is converted to general DEVS Java Model which contains required variables and functions for the DEVS formalism, together with users' added variables and functions. The State Designer enables state transitions for a general DEVS model to be characterized using the designations of Input Port, Output Port, No Output, and Probability as described in Table 1.

Table 1. State Designer User Specifications of Transition Types

Type	**User Specifies**	**Effect**
Input Port	name and type of message received	Enables an external event tag block for code handling of the messages received to implement the transition

Output Port	name and type of message generated	Enables an output event tag block for code generating the messages
No Output	N/A	Enables an internal event tag block for code implementing the transition
Probability	Probability of transition	Described in text

4.1.3 Probability Transition Specification

The Probability type is used to dynamically pick a next state among successor states with specified probabilities. In addition to having an input field to enter a probability value, the Probability type works similarly to the No Output type in that it enables an internal event tag block for code to implement the actions associated with the transition (called Finite Probability DEVS (FPDEVS)(Seo et al. 2015)).

The state diagram displays a different (dotted) line for a Probability type to distinguish it from other transition types (solid lines). Likewise, the DNL file needs to have two capabilities such as saving probability values for transitions and implementing a random selection for probability transitions. To preserve the probability values for transitions in a DNL file, the tag block in the internal transition function includes an XML document (Figure 6,) The document describes triples of the form (start state, end state, and probability value). To select a next state from such a specification, a function called *internalTransitionForFPDEVS* is placed in the tag block and defined in an additional code section in which customized function definitions can be inserted. The function is called in the internal transition function if the current state has Probability type transitions.

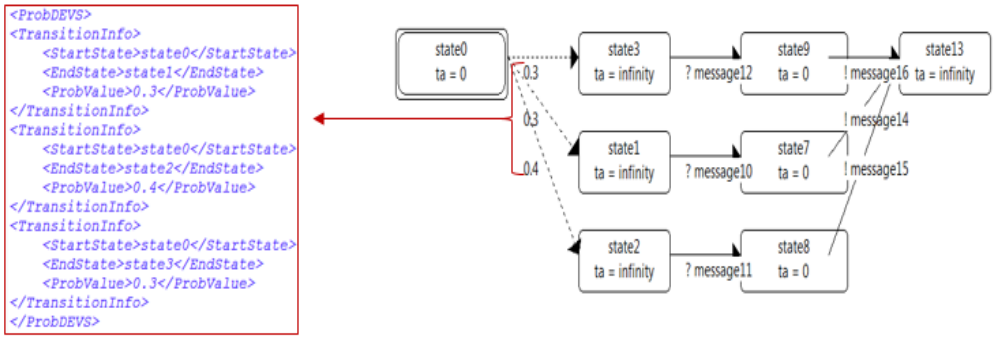

Figure 6: XML and state diagram of FPDEVS

The TransitionInfo tag shows each Probability type Transition whose information comprises start state, end state, and probability value recorded in StartState, EndState, and ProbValue tags, respectively.

Figure 6 illustrates a state diagram for FPDEVS and XML document containing Probability type transitions' information. Three dotted lines with probability values represent Probability type transitions from the state0. Solid lines show normal transition lines with question marks(?) and exclamation marks(!) before port names. The question mark is for an Input Port transition, and the exclamation mark for an Output Port transition. A shape having two rectangles represents the initial state in the state diagram. A state displays the name of the state and time advance (ta). The state diagram is converted to a DNL file which can later be restored in the state designer. For this purpose, the DNL file contains information of FPDEVS, especially Probability transition information in an associated XML document. The entries in the associated tags are used to construct Probability type transitions for the selected DNL file in the state diagram.

4.2 Constructing DEVS Markov Models in MS4 Me

With this background, we proceed to describe how state designer can be extended to support construction of Markov model types such as Continuous Time Markov (CTM), Discrete Time Markov(DTM), and Markov Matrix(MM). A Markov model is designed with Probability type transitions in the state diagram. A probability matrix from the Markov model is a key piece of information to execute the model in the MS4 Me environment. Each Markov type DEVS model has its own additional required variables, codes, and functions to execute its behaviors. For example, a CTM model decides its next state using probability values and sojourning time of the next state using a default or assigned distribution function. A DTM uses a fixed sojourning time for the current state, and a MM utilizes the probability matrix to calculate the equilibrium state vector. In contrast to a normal DEVS model, the Markov design in the state diagram is transformed to an XML document and a DNL file. Figure 7 illustrates how to create three Markov DEVS models using the state designer. To generate a CTM or DTM model, users first

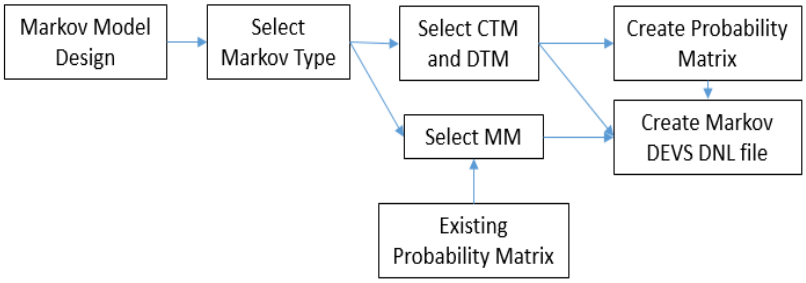

Figure 7: Markov Model Creation

construct a Markov transition model in the state diagram. Then on the first time through the process, they select Continuous Time Markov or Discrete Time Markov in the Create Markov Model dialog window. The Markov state designer generates an XML document which contains Probability type information, and a DNL file with supporting variables, codes, and functions for a CTM or DTM DEVS model. Subsequent modifications of the models only require incremental inputs which are saved to the created files. Once a probability matrix exists in an XML file, users can create a Markov Matrix model using it with the Create Markov Model dialog window.

4.2.1 Extended DEVS Markov Model

In a DEVS Markov model, a GUI to set time information for each transition is added to enable users' specification of transition time distribution functions.

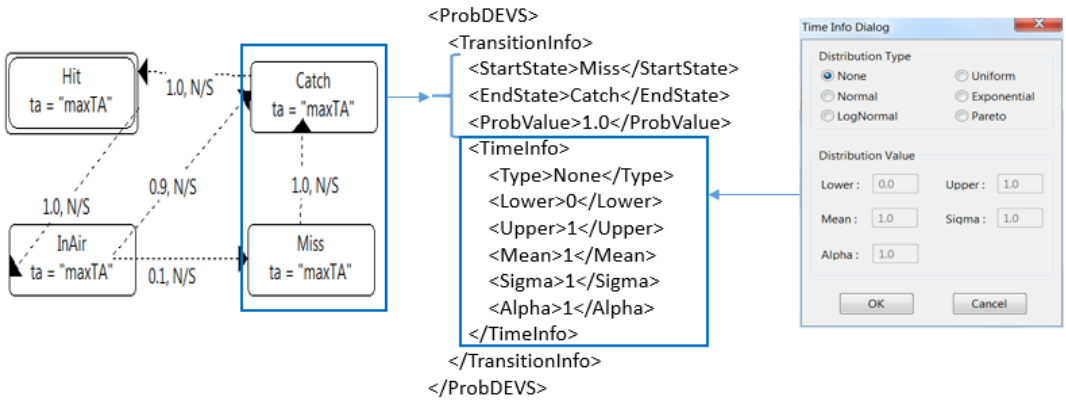

Figure 8: Generation of Time Information on Transition

Figure 8 shows how information in a TransitionInfo tag is created from the state diagram. As with FPDEVS, a transition displays a probability value but adds as additional information the mean value of the distribution function assigned by the user. The state diagram provides a Time Info Dialog window as seen in the right side of the Figure 8 to select five distribution functions with the None type as a default. (showing a transition line with N/S (Not Specified)). Each distribution function needs to have on or more parameters. For example, the Uniform distribution function requires upper and lower bounds as parameters, and Exponential distribution function requires only a mean value. The Time Info Dialog enables entry of parameter value(s) by selecting a distribution function type. The XML document is parsed to generate a list of TransitionInfo instances in the ContinuousTimeMarkov class which contains utility functions to perform the functions described in Section 3.2.

4.2.2 Packages for DEVS Markov Modeling and Simulation

- com.ms4systems.devs.markov
  - ContinuousTimeMarkov.java
  - TimeInState.java
  - TransitionInfo.java
  - TimeInfo.java
- com.ms4systems.devs.markovmodel
  - MarkovMat.java
- com.ms4systems.devs.analytics
  - SampleFromDistribution.java
  - SampleFromExponential.java
  - SampleFromLogNormal.java
  - SampleFromNormal.java
  - SampleFromPareto.java
  - SampleFromUniform.java

Figure 9: Packages for DEVS Markov Model

MS4Me provides three packages to support Markov DEVS models as seen in Figure 9. The ContinuousTimeMarkov class contains instances of other classes in the com.ms4systems.devs.markov package to hold all transition information and associated time information from an XML document. The TransitionInfo class has an instance of the SampleFromDistribution class which is a super class for all classes in the com.ms4systems.devs.analytics package. Based on a distribution type in a TimeInfo class, the TransitionInfo class can have instances of the five distribution types. For example, if a TimeInfo class has a Normal distribution type, the TransitionInfo class generates the instance of the SampleFromNormal class with the mean value from the TimeInfo instance. Each distribution class has a function to generate a random value which is used for transition time to a next state. If None distribution type is selected, the SampleFromExponential class is used as a default. The TimeInState class accumulates residence time in each state during simulation runs. The MarkovMat class is used in DEVS Markov Matrix models to calculate probability matrices.

DEVS Markov models have functions to help execute each Markov type. A CTM DEVS model uses an internalTransitionForMarkov function to sample a next state and transition time. The function is called whenever internal events occur in the CTM model. A DTM DEVS model has an internalTransitionForDiscreteMarkov function to compute a next state whenever internal transitions are triggered in the DTM DEVS model. In Markov Matrix, the internalTransitionForMatrix function is executed at each internal event.

5 DISCUSSION

5.1 Implementation of GSMP

Nielsen (1998) proposed a compositional GSMP modeling methodology, GMSim implemented in C++. Its close relation to the underlying mathematical structure facilitates coherent modeling and efficient implementation. Although claimed to be completely generic and extendible, the claim is based on the implementation software rather than on the explicit systems-theory framework provided by DEVS. Rachelson et al (2008) pointed out the need for including time as a basic variable in the formulation of Markov Decision Processes and emphasized the advantage of their implementation of GSMP within the DEVS-based Virtual Lab Environment (VLE). We note that the presented MS4 Me implementation has the same or more representation power as the VLE DEVS implementation of GSMP. Briefly put, the DEVS Markov coupled model is the resultant of coupling of atomic components with PDEVS semantics. (When

an atomic is imminent it picks a new state and a time from a distribution just like GSMP. It can send outputs to others atomics to change them. Unaffected atomics retain their time lefts just as GSMP clocks do.) Moreover the MS4 Me implementation is generic while the one in VLE is specific to the MDP problem addressed.

5.2 Computation of Behaviors

Simulation environments, including GMSim, VLE, and MS4 Me, generate the behavior of the resultant system (DEVS coupled model) based on the interaction of its components. In contrast, analytic approaches work with the state transition-rate-matrix of the resultant as the object of computation. (In MS4 Me, the Markov Matrix is automatically constructed in the state designer for atomic, but not for, coupled models.) In a well-known phenomenon, the state transition-rate-matrix of the resultant frequently explodes through the cross product of component state sets. Analytic computational methods have been developed that attempt to avoid such explosion (Bobbio et al 2016; Dayar 2013; Deavors and Sanders 1997; Banisch 2015). However, by not explicitly constructing the resultant matrix, simulation models are subject to linear, rather than exponential, growth. Moreover, they can employ parallel and distributed methods directly to coordinate computation of the model components and their message exchange. Finally, while analytic methods typically address performance metrics that can be computed from steady state solutions, simulations support a wide range of transient and steady state behaviors. Particularly, the full integration of DEVS Markov models with DEVS models of other classes, enables exploration of a virtually unlimited range of metrics and behaviors.

6 CONCLUSION

This paper has presented the concepts underlying DEVS Markov Modeling and Simulation and their implementation in MS4 Me. The Markov concepts of states and state transitions are fully compatible with the DEVS characterization of discrete event systems. They present a natural basis for the integrated Markov modeling facility built by extending the MS4 Me state designer and related concepts such as the DEVS natural language and its tag block feature. DEVS Markov atomic models are informed by probability and time transition structures which let them randomly pick next states and times of transition according to modeler specifications. Moreover, their inherited input/output capabilities (ports, external transition functions, output functions) establish them as full-fledged DEVS models able to be coupled with other DEVS components to construct complex hierarchical models. Also, due to their explicit transition and time advance structure, DEVS Markov models can be individualized with specific transition probabilities and transition times/rates which can be changed during model execution for dynamic structural change.

REFERENCES

Castro, R., Kofman E., and Wainer, G (2010) A Formal Framework for Stochastic DEVS Modeling and Simulation, Simulation J., Volume: 86 issue: 10, page(s): 587-611

Zeigler, B.P., Muzy, A., and Kofman, E. 2018.Theory of Modeling and Simulation: Discrete Event & Iterative System Computational Foundations, 3rd Ed., Elsevier Publisher, NY.

Zeigler, B.P. and P.E. Hammonds. 2007. Modeling & Simulation-Based Data Engineering: Introducing Pragmatics into Ontologies for Net-Centric Information Exchange.

M.H. Hwang and B.P. Zeigler. 2009. Reachability Graph of Finite and Deterministic DEVS Networks. IEEE Transactions on Automation Science and Engineering, Volume 6, Issue 3, pp.454-467.

Feller, W. 1966. "An introduction to probability theory and its applications", 1–2, Wiley.

Kemeny, J.G. and J.L. Snell, 1960, Finite Markov Chains, v. Nostrand.

Barbu V.S. and Limnios N. 2008. Semi-Markov Chains and Hidden Semi-Markov Models towards Applications, pp.1-10 et 45-61.

Rachelson, E., Quesnel, G., Garcia, F., and Fabiani, P. 2008. A Simulation-based Approach for Solving Generalized Semi-Markov Decision Processes. In European Conference on Artificial Intelligence.

Glynn, P. 1989. A GSMP Formalism for Discrete Event Systems. Proc. of the IEEE, 77.

Zeigler B. P., and Sarjoughian, H. S. 2017. "Guide to Modeling and Simulation of Systems of Systems". Springer; 2nd edition.

Bernard P. Zeigler, Chungman Seo, Robert Coop and Doohwan Kim. 2013. Creating Suites of Models with System Entity Structure: Global Warming Example. SpringSim (TMS-DEVS).

Chungman Seo, Bernard P. Zeigler, Robert Coop and Doohwan Kim. 2013. DEVS Modeling and Simulation Methodology with MS4Me Software Tool. SpringSim (TMS-DEVS).

Bobbio, A., et al. 2016 Markovian Agent Models: A Dynamic Population of Independent Markovian Agents. In Seminal Contributions to Modeling and Simulation, Eds: K. Al-Begain and A. Bargiela, Springer.

Dayar, T. 2013. Analyzing Markov Chains using Kronecker Products: Theory and Applications, Springer, Briefs in Mathematics, Berlin, Germany, DOI: 10.1007/978-1-4614-4190-86.

Banisch Sven. 2015. Markov Chain Aggregation for Agent-Based Models. pp 35-55. in Agent-BasedModels as Markov Chains Springer.

Nielsen F. 1998. GMSim: a tool for compositionnal GSMP modeling. In Winter Simulation Conference.

D.D. Deavours ; W.H. Sanders. 1997."On-the-fly" solution techniques for stochastic Petri nets and extensions ,Proceedings of the Seventh International Workshop on Petri Nets and Performance Models.

Puterman M. 1994. Markov Decision Processes. John Wiley & Sons, Inc.

Younes H. K: Simmons R. 2004. Solving generalized semi-markov decision processes using continuous phase-type distributions. In AAA I.

Seo, C., Zeigler, B.P., Kim, D. and Duncan, K. 2015. "Integrating Web-based Simulation on IT Systems with Finite Probabilistic DEVS", In *Proceeding of the Symposium on TMS: DEVS Integrative M&S Symposium (TMS-DEVS)*.

Zeigler, B.P., Seo, C. and Kim, D. 2013. *System Entity Structures for Suites of Simulation Models*, International Journal of Modeling, Simulation, and Scientific computing, Volume 04, Issue 03.

AUTHOR BIOGRAPHIES

Chungman Seo is a senior research engineer at RTSync Company and a member of Arizona Center for Integrative Modeling & Simulation (ACIMS). He received his Ph.D. in Electrical and Computer Engineering from The University of Arizona in 2009. His research includes DEVS based web service integration, DEVS/SOA based distributed DEVS simulation, and DEVS simulator interoperability. His email address is cseo@rtsync.com.

Bernard P. Zeigler is Chief Scientist at RTSync Corp., Professor Emeritus of Electrical and Computer Engineering at the University of Arizona, Tucson and co-Director of the Arizona Center for Integrative Modeling and Simulation. His email address is zeigler@rtsync.com.

Doohwan Kim is the founder and president of RTSync and MS4 Systems. He is the first to introduce the commercial innovation of the DEVS modeling and simulation software toolsets and Cloud based model store environment. He received his Ph.D. in Electrical and Computer Engineering from the University of Arizona in 1996. His email address is dhkim@rtsync.com.