A Formal Framework for Stochastic Discrete Event System Specification Modeling and Simulation

Rodrigo Castro, Ernesto Kofman and Gabriel Wainer

SIMULATION 2010 86: 587 originally published online 29 June 2009
DOI: 10.1177/0037549709104482

A Formal Framework for Stochastic Discrete Event System Specification Modeling and Simulation

Rodrigo Castro
Ernesto Kofman
Universidad Nacional de Rosario, Riobamba 245 bis,
Rosario 2000, Santa Fe,
Argentina
rodrigocastro@ieee.org
kofman@fceia.unr.edu.ar

Gabriel Wainer
Carleton University, 1125 Colonel By Drive,
Ottawa, ON, K1S 5B6 Canada
gwainer@sce.carleton.ca

We introduce an extension of the classic Discrete Event System Specification (DEVS) formalism that includes stochastic features. Based on the use of the probability spaces theory we define the stochastic DEVS (STDEVS) specification, which provides a formal framework for modeling and simulation of *general non-deterministic* discrete event systems. The main theoretical properties of the STDEVS framework are treated, including a new definition of *legitimacy* of models in the stochastic context and a proof of STDEVS *closure under coupling*. We also illustrate the new stochastic *modeling* capabilities introduced by STDEVS and their relation with those found in classic DEVS. Practical *simulation* examples are given involving performance analysis of computer systems and hybrid modeling of networked control systems, applications where the modeling of stochastic components is *vital*.

Keywords: Discrete Event Simulation, Stochastic Systems, DEVS, Hybrid System Modeling

1. Introduction

DEVS (Discrete Event System Specification) is a formalism that was developed in the mid-1970s [1, 2] as a general methodology for describing discrete event systems. DEVS is a system theoretic-based representation of the systems whose input/output behavior can be described by sequences of events. Being a universal formalism for discrete event systems [3], other methodologies (e.g. finite state automata, Petri nets, grafcets, statecharts, etc.) can be represented by DEVS. This generality converted DEVS into a widely used formalism to describe and to simulate most classes of discrete systems, including discrete time systems [2]. Moreover, numerical integration methods that approximate continuous systems (differential equations) by DEVS models have been developed [4] and several applications and extensions of the DEVS formalism for modeling and simulation of continuous and hybrid systems have been proposed [5, 6].

Many DEVS-based modeling and simulation software tools have been developed through the years [7–10]. Although most of these tools have incorporated the use of random functions, DEVS has only been formally defined for modeling deterministic systems, which limits the extent of the formal framework to a wide family of stochastic systems.

This work introduces a general DEVS-based formalism for modeling generalized stochastic systems. The new formalism, called STDEVS (*stochastic DEVS*) is an ex-

SIMULATION, Vol. 86, Issue 10, October 2010 587–611

DOI: 10.1177/0037549709104482
Figures 2, 5, 7, 8 appear in color online: http://sim.sagepub.com

tension of DEVS that establishes a formal framework for modeling and simulation of stochastic discrete event systems. The main theoretical properties of STDEVS (such as closure under coupling, legitimacy, etc.) are formally defined in the context of the new framework.

Stochastic models play a fundamental role in discrete event system theory. In fact, any system involving uncertainties, unpredictable human actions or system failures requires a non-deterministic treatment. Examples of traditional stochastic discrete event formalisms are Markov chains [11], queueing networks [12] and stochastic Petri nets [13]. These techniques permit stochastic models to be analyzed and simulated in several applications.

Some early works have dealt with the links between DEVS and stochastic systems [14–16]. Nevertheless, none of them provided a general theory or a formal theoretic support for modeling general stochastic DEVS models

The STDEVS methodology, besides providing a mechanism to specify general stochastic DEVS models, describes the probabilistic behavior from a *state transition specification level* and covers coupling properties in a straight manner. STDEVS inherits the DEVS multimodeling capabilities to represent hybrid systems, and offers enhanced simulation performance compared with traditional techniques when combining stochastic discrete time and continuous time systems.

The work is organized as follows. Section 2 provides the required background about DEVS, previous attempts to link DEVS and stochastic phenomena, and the probability spaces theory (which we use to define STDEVS in terms of general state sets). Section 3 introduces several motivating modeling problems and formulates examples to make evident the main limitations of the classic DEVS framework to represent general stochastic systems. Section 4 provides the rationale behind the proposition of STDEVS and defines the STDEVS formalism. Section 5 shows that STDEVS is closed under coupling, and it defines the property of legitimacy. Section 6 shows that any measurable DEVS model where the transition functions depend on random variables defines an equivalent STDEVS model (which permits some STDEVS models to be modeled without making use of probability spaces and provides a formal framework for DEVS simulation tools using pseudo-random sequence generators). The section ends with a summary and discussion that substantiates the benefits of STDEVS by solving the motivating problems posed in Section 3. Finally, Section 7 illustrates the use of the new formalism in the context of a stochastic modeling process, using different examples in computer and networking applications that involve hybrid (continuous/discrete) modeling and control theory.

2. Background

2.1 DEVS Formalism

As we said earlier, the DEVS formalism can describe most discrete systems, including discrete time systems and, more recently, continuous and hybrid systems. This generality is achieved by the ability of DEVS to represent general *discrete event systems*, i.e. any system whose input/output behavior can be described by sequences of events.

More specifically, a DEVS model [2] processes an input event trajectory and, according to that trajectory and to its own initial state, provokes an output event trajectory. Formally, a DEVS *atomic* model is defined by the following structure:

$$M = (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta),$$

where

- X is the set of input event values, i.e. the set of all the values that an input event can take;
- Y is the set of output event values;
- S is the set of state values;
- δ_{int}, δ_{ext}, λ and ta are functions which define the system dynamics.

Each possible state s ($s \in S$) has an associated *time advance* calculated by the *time advance function* $ta(s)$ ($ta(s) : S \to \Re_0^+$). The *time advance* is a non-negative real number saying how long the system remains in a given state in absence of input events.

Thus, if the state adopts the value s_1 at time t_1, after $ta(s_1)$ units of time (i.e. at time $ta(s_1) + t_1$) the system performs an *internal transition*, going to a new state s_2. The new state is calculated as $s_2 = \delta_{\text{int}}(s_1)$, where δ_{int} ($\delta_{\text{int}} : S \to S$) is called the *internal transition function*.

When the state goes from s_1 to s_2 an output event is produced with value $y_1 = \lambda(s_1)$, where λ ($\lambda : S \to Y$) is called *output function*. Functions ta, δ_{int} and λ define the autonomous behavior of a DEVS model.

When an input event arrives, the state changes instantaneously. The new state value depends not only on the input event value but also on the previous state value and the elapsed time since the last transition. If the system goes to the state s_3 at time t_3 and then an input event arrives at time $t_3 + e$ with value x_1, the new state is calculated as $s_4 = \delta_{\text{ext}}(s_3, e, x_1)$ (note that $ta(s_3) \geq e$). In this case, we say that the system performs an *external transition*. Function δ_{ext} ($\delta_{\text{ext}} : S \times \Re_0^+ \times X \to S$) is called the *external transition function*. No output event is produced during an external transition.

DEVS models can be coupled in a modular way [2]. A DEVS coupled model N is defined by the structure:

$$N = (X_N, Y_N, D, \{M_d\}, \{I_d\}, \{Z_{i,d}\}, \mathit{Select})$$

where:

- X_N and Y_N are the sets of input and output values of the coupled model;
- D is the set of component references, so that for each $d \in D$, M_d is a DEVS model;
- for each $d \in D \cup \{N\}$, $I_d \subset (D \cup \{N\}) - \{d\}$ is the set of influencer models on subsystem d;
- for each $i \in I_d$, $Z_{i,d}$ is the translation function, where

$$Z_{i,d} : \begin{cases} X_N \to X_d & \text{if } i = N \\ Y_i \to Y_N & \text{if } d = N \\ Y_i \to X_d & \text{otherwise} \end{cases}$$

- *Select*: $2^D \to D$ is a tie-breaking function for simultaneous events; it must verify $Select(E) \in E$, with $E \subset 2^D$ the set of components producing the simultaneity of events.

DEVS models are closed under coupling, i.e. the coupling of DEVS models defines an equivalent atomic DEVS model [2].

2.2 Early Links between DEVS and Stochastic Phenomena

From an early stage it was clear that there is a need to establish formal relationships to allow DEVS to describe stochastic characteristics of the systems under study. In [14, 1] the authors showed that a discrete event simulation driven by pseudo-random sequences defines an equivalent DEVS model. In other words, the DEVS formalism can capture the behavior of stochastic systems that are being simulated using pseudo-random sequences, i.e. they perform *deterministic modeling of stochastic systems* [1]. However, it does not provide a methodology to describe DEVS stochastic models. In [15] the author established a relationship between probabilistic experiment outcomes and the evolution of a DEVS simulation, by assigning stochastic measures to the externally observable state trajectories. However, this work is limited to models described at the input/output level of specification [2] (i.e. it does not specify the dynamics at the state transition level).

The first structural approach to extend DEVS to take into account an internal stochastic behavior at the state transition level was sketched in [16]. However, that work was limited to DEVS models with finite state sets. Since one of the most important features of DEVS is its capability to deal with arbitrary state sets, the restriction to finite sets is a problem. These works led to a set of useful results that were sufficient to tackle some specific problem domains, but only from a behavioral perspective.

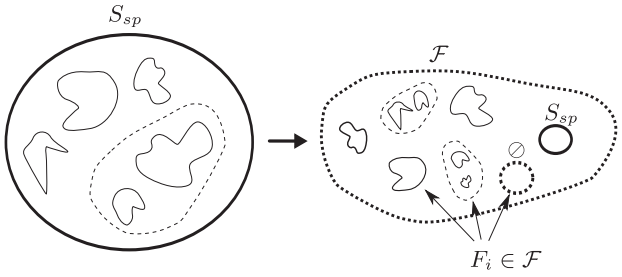

Figure 1. A sample space S_{sp} and its generated sigma-field $\mathcal{F}$. Each member F_i of the collection $\mathcal{F}$ satisfies the required conditions imposed to constitute a sigma-field. The empty space $\oslash$ and the original sample space S_{sp} are required members of $\mathcal{F}$.

2.3 Probability Spaces

As we will see, to overcome this difficulty, we define STDEVS in terms of general probability spaces, relying on the general theory of probability spaces [17, 18].

A *sample space* S_{sp} of a random experiment is a set that includes all of the possible outcomes of the experiment. A *sigma-field* (also referred as a *sigma-algebra*[1]) $\mathcal{F}$ of the sample space S_{sp} is a non-empty collection made of subsets of S_{sp}. The idea is that, when we work with a continuous sample space S_{sp}, we can assign probabilities to subsets of S_{sp} and not to single elements of S_{sp}. Thus, the probabilities are assigned to the elements of the sigma-field $\mathcal{F}$ (which are subsets of S_{sp}).

A sigma-field cannot be any arbitrary collection of subsets of S_{sp}. A collection $\mathcal{F}$ must satisfy the following properties in order to constitute a sigma-field:

- if $F \in \mathcal{F}$, then $F^c \in \mathcal{F}$ (where F^c is the complement of F in S_{sp}, $F^c \cup F = S_{\text{sp}}$);
- if $F_i \in \mathcal{F}$ for $i = 1, \ldots, \infty$, then also $\bigcup_{i=1}^{\infty} F_i \in \mathcal{F}$.

Note that since $F^c \cup F = S_{\text{sp}}$, the last two conditions imply that $S_{\text{sp}} \in \mathcal{F}$ and also $\oslash \in \mathcal{F}$. In Figure 1 we show a schematic representation of a sigma-field generated from an arbitrary sample space and its members.

These conditions are necessary in order to build up a general measurable structure from S_{sp}, which can be equipped with probability measures as we show shortly.

One particular example of a sigma-field over the sample space S_{sp} is the collection of all of the possible subsets of S_{sp} ($2^{S_{\text{sp}}}$, called the *power set* of S_{sp}). Although this is not a very useful sigma-field (as it may include many uninteresting subsets, to which we may not want or may not be able to assign probabilities), it helps in many theoretical proofs.

1. A sigma-algebra is also called an *event space*, but we avoid this term in order not to become confused with the meaning we give to the word *event* in the DEVS methodology.

In most practical problems, we would only describe the probabilities of *certain subsets* of S_{sp} (e.g. if S_{sp} was the real set $\Re$, we normally assign probabilities only to open intervals). Although an arbitrarily chosen collection of these subsets (that may be of interest for working on a particular problem) may not constitute a sigma-field itself, it always *generates* a sigma-field. Let $\mathcal{G}$ be a particular collection of subsets of S_{sp}. The sigma-field generated by $\mathcal{G}$, denoted $\mathcal{M}(\mathcal{G})$, is the smallest sigma-field that contains all of the elements of $\mathcal{G}$.

A pair $(S_{sp}, \mathcal{F})$ consisting on a sample space S_{sp} and a sigma field $\mathcal{F}$ of subsets of S_{sp} is called a *measurable space*. A *probability measure* P on a measurable space $(S_{sp}, \mathcal{F})$ is an assignment of a real number $P(F)$ to every member F of the sigma-field, such that P obeys the following rules.

- *Axiom 1*: $P(F) \geq 0$ for all $F \in \mathcal{F}$ (the probabilities are non-negative).
- *Axiom 2*: $P(S_{sp}) = 1$ (the probability of an outcome in the complete sample space is equal to 1).
- *Axiom 3*: if $F_i \in \mathcal{F}, i = 1, \ldots, \infty$ are disjoint sets, then $P(\bigcup_{i=1}^{\infty} F_i) = \sum_{i=1}^{\infty} P(F_i)$.

When we have a sigma-field $\mathcal{F} = \mathcal{M}(\mathcal{G})$ generated from a particular practical collection $\mathcal{G}$ of subsets of S_{sp}, the knowledge of $P(G)$ for every subset $G \in \mathcal{G}$, readily defines the function P for every subset $F \in \mathcal{F}$.

Finally, our random experiment can be fully described by a *probability space* defined as the triplet $(S_{sp}, \mathcal{F}, P)$ consisting of a sample space S_{sp}, a sigma-field $\mathcal{F}$ of subsets of S_{sp}, and a probability measure P defined for all members of $\mathcal{F}$.

Synthesizing, for every $F \in \mathcal{F}$, $P(F)$ expresses the probability that the random experiment produces a sample $s \in F \subseteq S_{sp}$ as the experiment outcome.

2.4 Measurable Functions

We also need to use the concept of *measurable functions* [18] to describe certain requisites for the model functions. The concept is described through the following definitions.

Definition 1. (Preimage) *Given a function $f : A \to B$ and the set $B_1 \subseteq B$, we define the* preimage $A_1 \triangleq f^{-1}(B_1)$ *of the set B_1 under f as*

$$A_1 = \{a \in A \mid f(a) \in B_1\}.$$

Definition 2. (Measurable function) *Given a sigma-field $\mathcal{A}$ over a set A and a sigma-field $\mathcal{B}$ over a set B, a function $f : A \to B$ is said to be* measurable $\mathcal{A}/\mathcal{B}$ *when it holds:*

$$A_1 = f^{-1}(B_1) \Rightarrow A_1 \subseteq A, \quad \forall B_1 \subseteq B.$$

The idea is that when a measurable function renders a measurable image set it ensures a measurable preimage set. These sets are measurable provided that they belong to their corresponding sigma-fields, and the function itself is regarded *measurable* in the context of $\mathcal{A}$ and $\mathcal{B}$. For brevity, in general the reference to sigma-fields is omitted, and f is simply called a *measurable function*.

3. Motivation

In this section we expose the main theoretical issues that raise the need for a reconsideration of the DEVS formalism for modeling stochastic processes. We discuss a series of open questions that show the need for a new approach to solve this problem.

3.1 DEVS with Random Numbers: DEVS-RND

We start with the following basic question: can random number generators be used in DEVS functions?

We study some very simplistic DEVS models that incorporate the use of random numbers, and find that it is not difficult to quickly reach very important limitations that invalidate the correctness of the models. In the following, we call those DEVS models whose functions depend on (at least one) random numbers DEVS-RND models, and focus mainly on the internal transition function to build our cases. Let us start with a basic DEVS-RND model M_U that throws a uniformly distributed random number between zero and one on its output every second. Such a model can be defined as follows:

$$M_U = (X, Y, S, \delta_{int}, \delta_{ext}, \lambda, ta)$$

where:

- $X = \oslash$ (the model has no inputs);
- $Y = \Re_{[0,1]}$;
- $S = \Re_{[0,1]} \times \Re_0^+$;
- $\delta_{int}(a, \sigma) = (\text{RND}(0, 1), 1)$ (always one internal transition per second);
- $\delta_{ext}(s, e, x) = \oslash$ (the model has no external transitions);
- $\lambda(s) = a$ (where $a \in \Re_{[0,1]}$ is the part of the state used to store the last generated random value);
- $ta(s) = \sigma$ (where $\sigma \in \Re_0^+$ is the part of the state used to store the time to the next internal transition).

Now suppose that at a given time the model state is $s = (0.5, 1)$. The next state s' will be determined by $s' = \delta_{int}(0.5, 1)$ which is an undetermined value until the random experiment described by RND(0, 1) is evaluated.

This situation shows that δ_{int} *is not a function* (it does not assign a unique value to a given set of input variables). So, the model is not a proper DEVS model (δ_{int} must be a function according to its formal definition).

The workaround to this simple situation is to rewrite[2] $\delta_{\text{int}}(\cdot)$ with a proper structure that mathematically reflects a function, yet describing the random behavior. (In the work of [1] where δ_{int} was first defined formally, a *grocery store* example is provided where a function $\Gamma : [0, 1] \rightarrow [0, 1]$ is used as a random number generator to build an internal transition function such as $\delta_{\text{int}}(s) = f(\cdot, \Gamma(s))$. This is a proper DEVS definition given $\Gamma(\cdot)$ represents a case of *deterministic modeling* of a stochastic system. Strictly speaking, δ_{int} thus defined does not depend on a true random variable, but instead on a deterministic trajectory of values Γ. This trajectory is *intended to mimic* the statistical properties of a *real* random process being modeled (i.e. a pseudo-random sequence). With the purpose of accurate modeling representativeness, function Γ is required to provide the '*notion of an* ideal *random number generator*', and the guidelines to formulate this requirement in formal terms are provided in [1]. Nevertheless, δ_{int} remains deterministic, thus complying with its mathematical definition as a function.)

This is achieved by incorporating the random variable as an argument. With this alternative approach, we can write

$$\delta_{\text{int}}(a, \sigma, r) = (r, 1) \quad \text{with } r \sim U(0, 1)$$

In general, we may say that now the internal transition function depends on the state *s and* some random variable r with a given statistical distribution, and that this is mathematically sound.

Working in that way, the function $\delta_{\text{int}}(s, r)$ computes a state $s \in S$ according to the choice of a random number r.

3.2 Generality of DEVS-RND

Now that we have our DEVS-RND form with a well-defined internal transition function, we pose the next question: is the form $\delta_{\text{int}}(s, r)$ the most general way of choosing an element out of set S?. The answer is no.

Since S in DEVS is an abstract set, it can have a very complex structure so that real numbers cannot be uniquely mapped into elements of S. For instance, we can think of a DEVS-RND model that every second randomly chooses a function $f_c : [0, 1] \rightarrow \Re$ from the space of all continuous functions $C([0, 1])$, with $f_c \in C([0, 1])$. Then, it calculates the average value of f_c over the [0, 1] interval and sends it as an output value. In such a model, the state space is infinite-dimensional, and given the fact that real numbers cannot be mapped into this space, the model cannot be defined using the DEVS-RND structure. The only general way to randomly choose elements from the space of continuous functions is through the use of probability spaces. Although for most practical situations in simulation choosing real numbers would suffice, our goal is to develop a general modeling formalism.

3.3 Consistency of DEVS-RND

We proceed now with the next question: is DEVS-RND consistent? The idea is to know whether or not it is consistent[3] in DEVS-RND to add random variables as arguments as we did in the function δ_{int}.

To answer this question, we introduce a second example. The following model M_V chooses a random number $r \sim U(0, 1)$ every second in the same way as model M_U does, but this time M_V also computes the average number of times the outcome of r belongs to a given subset $V \subset \Re_{[0,1]}$. The new model looks as follows:

$$M_V = (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta)$$

where:

- $X = \oslash$ (the model has no inputs);
- $Y = \Re_{[0,1]}$;
- $S = \Re_{[0,1]} \times \mathbb{N} \times \Re_0^+$;
- $\delta_{\text{int}}(v, n, \sigma, r) = (v', n + 1, 1)$ with

$$v' = \begin{cases} \frac{v \cdot n + 1}{n+1}, & \text{if } r \in V \subset \Re_{[0,1]} \\ \frac{v \cdot n}{n+1}, & \text{otherwise} \end{cases}$$

  (where n is the part of the state used to count the number of experiments);

- $\delta_{\text{ext}}(s, e, x) = \oslash$ (the model has no external transitions);
- $\lambda = v$ (where $v \in \Re_{[0,1]}$ is the part of the state used to store the average number of successes for $V \subset \Re_{[0,1]}$);
- $ta = \sigma$ (where $\sigma \in \Re_0^+$ is the part of the state used to store the time to the next internal transition).

It is clear that as n goes to infinity the output value converges to the probability that a real number between zero and one belongs to V. However, if V was a Vitali set [19], that probability does not exist, since sets of the Vitali type are non-measurable sets. A Vitali set is an example of a set of real numbers that is not Lebesgue measurable, being *inconsistent* to talk about its length.

2. We use the symbol $\cdot$ in the argument of a function to denote that the given function depends on *some list of arguments*.

3. We refer to the concept of *consistency* informally, to denote the situation when a model does not lead to a mathematical contradiction.

Hence, although the model M_V looks correct according to our *solution* of adding r into the arguments of δ_{int}, it is mathematically inconsistent. We cannot generate a random real number and then ask whether it belongs to a Vitali set, as this set has no measure on the real numbers.

Thus, the answer to our question is again no. We can find inconsistent models when adding random variables in the arguments of the transition functions.

3.4 Closure Under Coupling of DEVS-RND

Now we start checking DEVS-RND regarding the fundamental theoretical properties of classical DEVS, starting with closure under coupling. So our next question is: does the coupling of consistent DEVS-RND models always define an equivalent consistent DEVS-RND model?

In other words, we ask whether or not the closure under coupling can be still guaranteed when random variables are added to a DEVS model.

Consider again the model M_U introduced in Section 3.1, expressed according the DEVS-RND form:

$$M_U = (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta)$$

where:

- $X = \oslash$ (the model has no inputs);
- $Y = \Re_{[0,1]}$;
- $S = \Re_{[0,1]} \times \Re^{+}$;
- $\delta_{\text{int}}(a, \sigma, r) = (r, 1)$ with $r \sim U(0, 1)$ (one internal transition per second);
- $\delta_{\text{ext}}(s, e, x) = \oslash$ (the model has no external transitions);
- $\lambda(s) = a$;
- $ta(s) = \sigma$.

Now let us connect the output of M_U to the input of a deterministic DEVS model M_A, which computes in its state the average of the received values when they belong to the Vitali set $V \subset \Re_{[0,1]}$:

$$M_A = (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta)$$

where:

- $X = \Re_{[0,1]}$;
- $Y = \Re_{[0,1]}$;
- $S = \Re_{[0,1]} \times \mathbb{N} \times 0, \infty$;
- $\delta_{\text{int}}(v, n, \sigma) = (v, n, \infty)$;
- $\delta_{\text{ext}}(v, n, \sigma) = (v', n + 1, 0)$ with

$$v' = \begin{cases} \frac{v \cdot n + 1}{n+1}, & \text{if } r \in V \subset \Re_{[0,1]} \\ \frac{v \cdot n}{n+1}, & \text{otherwise} \end{cases}$$

- $\lambda(v, n, \sigma) = v$;
- $ta(v, n, \sigma) = \sigma$.

Both DEVS models are perfectly consistent. The first is very simple, and the second is just a classic deterministic DEVS model. However, the coupling of the consistent models M_U and M_A results in a model that behave in exactly same way as the M_V atomic model studied in Section 3.3, which is indeed inconsistent.

Thus, the answer to our question is no; working in this way we cannot guarantee closure under coupling while preserving consistency.

3.5 Legitimacy of DEVS-RND

Another important theoretical property of classic DEVS is about legitimacy. Thus, our next question is: can we apply the concept of legitimacy in classic DEVS to DEVS-RND models?

Consider now the following DEVS-RND model, which simply throws a value of one on its output on a randomly chosen time basis:

$$M_L = (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta)$$

where:

- $X = \oslash$ (the model has no inputs);
- $Y = 1$;
- $S = \Re_{[0,1]}$;
- $\delta_{\text{int}}(\sigma, r) = r$ with $r \sim U(0, 1)$;
- $\delta_{\text{ext}} = \oslash$ (the model has no external transitions);
- $\lambda(s) = 1$;
- $ta(s) = \sigma$.

Is this model legitimate, i.e. will it always perform a finite number of events in a given finite interval of time?

Note that the following sequence of values of r is feasible and results in an illegitimate model state trajectory: $\{1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \ldots\}$. Also, the sequence $\{0, 0, 0, 0, \ldots\}$ can occur, leading our model to a illegitimate condition.

However, we can easily prove that the *probability* of obtaining an illegitimating sequence is zero, thus obtaining a legitimate model *in some way*. However, to affirm this fact, we redefine the concept of legitimacy in the context of stochastic models.

3.6 The Need for a New Formalism

Now we reach to the last question: do we need a new formalism to define stochastic DEVS models?

The simple examples examined before in this section showed that adding stochastic features to DEVS models may lead to inconsistencies.

Other discrete event formalisms (such as Petri nets, finite state automata, statecharts, etc.) work with finite (or countable) state sets. Thus, adding stochastic features to them is straightforward.

DEVS, however, admits arbitrary state sets. Unfortunately, the only way of consistently working with random processes over arbitrary sets is through the use of probability spaces. All of the problems we saw in this section were a result of the fact that we tried to add stochastic features without taking care of the generality of the state space.

Thus, the answer to the last question is yes. It is clear that a general DEVS-based stochastic formalism must be provided if we propose to incorporate elements from the probability spaces theory. We introduce this new formalism, namely STDEVS, in the next section.

In practice, computers use a finite number of bits for data representation. Thus, when we simulate a DEVS model, the state set S is represented as a finite set (avoiding the problems discussed in this section). However, as DEVS distinguishes modeling from simulation, we should not make any assumption about the finiteness of the state space (which is a particular consequence of the simulation implementation).

Wrapping up, the benefit of a new STDEVS formalism is to provide the missing theory for obtaining a general, consistent, closed, legitimate and sound description for stochastic DEVS models.

4. STDEVS Formalism

This section introduces the new STDEVS formalism. After an informal description of the main idea, it presents the formal definition of an STDEVS atomic model and that of a coupled STDEVS model.

4.1 Concepts

Our basic approach is to take the deterministic DEVS definition (keeping the essence of its model structure), and then to derive a new stochastic model structure replacing the way dynamics are described. Namely, the deterministic functions δ_{int} and δ_{ext} should be replaced by their probabilistic counterparts in terms of the probability spaces theory. The new stochastic model structure will be referred to as *STDEVS* (which stands for *ST*ochastic *DEVS*), and guarantees the correctness of the usage of random-like functions into the model description and the coupling between stochastic and deterministic models.

We use the *transition* functions to incorporate stochastic behavior, thinking of each state transition event as a random experiment of which the possible outcomes will determine the next system state. As in DEVS internal transitions the future state $\tilde{s}$ is determined by the current state s, we define a probability space for each current state s (i.e. we assign probabilities to the transitions from s to future states). Similarly, in DEVS external transitions, the future state $\tilde{s}$ is determined by the current state s, the elapsed time e and the input event x. Thus, in STDEVS we define a probability space for each triplet (s, e, x) (i.e. we assign probabilities to the transitions from the triplet (s, e, x) to future states).

Both DEVS transitions and STDEVS stochastic experiments give $\tilde{s} \in S$ as a state transition result. Thus, the natural link between both types of structures (deterministic DEVS and stochastic STDEVS) is the set S. The set S acts as the *state set* for both structures, and as the *sample space* S_{sp} required to build a stochastic description in terms of probability spaces in the context of the STDEVS structure.

As the state set S can be continuous, a state-to-state probability measure might be useless, because it will be equal to zero for most possible future states $\tilde{s}$. When S is continuous, we need a continuous sample space and a continuous probability distribution to measure the probabilities of the future possible elements of S. This description is given by probability density functions (pdfs) which need to be integrated over an interval to produce the probability measure for the event that a random experiment outcome will land on that interval. In this scenario, if we choose an individual future state $\tilde{s}$ and calculate the integral of the pdf over a point, the calculation will render always zero. Thus, the general approach must consider probability measures describing the likelihood for the system being in state s arriving to *sets* $G \subset S$ of future states (rather than to a single element $\tilde{s} \in S$) after a model internal transition (and analogously for the case of the external transition).

In order not to lose generality, each transition in STDEVS must be seen as an independent random experiment. Thus, for an internal transition, depending on the current state, we should be able to assign probabilities for the future state belonging to different subsets of S. In other words, the collection $\mathcal{G}$ that contains the subsets of S to which we assign probabilities when the current state is s must depend on s. Thus, we define an internal set-collecting function $\mathcal{G}_{\text{int}}(s)$ which, together with the sample space S and a probability function define a probability space that replaces the DEVS internal transition function. A similar remark can be made for the external transition, where the set-collecting function $\mathcal{G}_{\text{ext}}(s, e, x)$ will also depend on the elapsed time and the input event.

Our approach relies on using the stochastic description *only to the state transition functions*. This allows us to greatly reduce description complexity for the stochastic structure, without losing generality. Randomness in func-

tions λ and ta, can be incorporated by taking into account that these functions depend on the state $s \in S$. Since s is always chosen by random experiments at the internal and external transition functions, the randomness of the remaining functions can be modeled as part of the information stored in the state. A straightforward way of doing this is to define the state as $s = (\hat{s}, \sigma, \ell)$ and then to make $ta(s) = \sigma$ and $\lambda(s) = \ell$. Then, we can define an arbitrary random experiment for choosing σ and ℓ at the internal and external transitions, and we obtain an arbitrary random behavior for the functions $\lambda(s)$ and $ta(s)$ (in spite of their deterministic definition).

4.2 STDEVS Definition

An STDEVS model has the structure:

$$M_{\text{ST}} = (X, Y, S, \mathcal{G}_{\text{int}}, \mathcal{G}_{\text{ext}}, P_{\text{int}}, P_{\text{ext}}, \lambda, ta)$$

where:

- X, Y, S preserve the original definition they have in DEVS;
- $\mathcal{G}_{\text{int}}, \mathcal{G}_{\text{ext}}, P_{\text{int}}, P_{\text{ext}}$ are new functions that replace the functionality of the original δ_{ext} and δ_{int} DEVS functions;
- λ, ta extend the original definition they have in DEVS with the additional requirement that they have to be *measurable functions*.

Here $\mathcal{G}_{\text{int}} : S \to 2^S$ is a function that assigns a collection of measurable sets $\mathcal{G}_{\text{int}}(s) \subseteq 2^S$ to every state s. Given a state s, the collection $\mathcal{G}_{\text{int}}(s)$ contains all of the measurable subsets of S that the future state might belong to with a known probability, determined by the function $P_{\text{int}} : S \times 2^S \to [0, 1]$. When the system is in state s the probability that the internal transition carries it to a set $G \in \mathcal{G}_{\text{int}}(s)$ is computed by $P_{\text{int}}(s, G)$.

Calling $\mathcal{F}_{\text{int}}(s) \triangleq \mathcal{M}(\mathcal{G}_{\text{int}}(s))$ to the minimum sigma-field generated by $\mathcal{G}_{\text{int}}(s)$, the triplet $(S, \mathcal{F}_{\text{int}}(s), P_{\text{int}}(s, \cdot))$ is a probability space for each state $s \in S$.

In a similar way, $\mathcal{G}_{\text{ext}} : S \times \Re_0^+ \times X \to 2^S$, is a function that assigns a collection of sets $\mathcal{G}_{\text{ext}}(s, e, x) \subseteq 2^S$ to each triplet (s, e, x). Given a state s and an elapsed time e, if an event with value x arrives, $\mathcal{G}_{\text{ext}}(s, e, x)$ contains all of the measurable subsets of S that the future state can belong to, with a known probability calculated by $P_{\text{ext}} : S \times \Re_0^+ \times X \times 2^S \to [0, 1]$.

Calling $\mathcal{F}_{\text{ext}}(s, e, x) \triangleq \mathcal{M}(\mathcal{G}_{\text{ext}}(s, e, x))$ to the minimum sigma-field generated by $\mathcal{G}_{\text{ext}}(s, e, x)$, the triplet $(S, \mathcal{F}_{\text{ext}}(s, e, x), P_{\text{ext}}(s, e, x, \cdot))$ is a probability space for every triplet (s, e, x).

4.3 Coupling in STDEVS

Proceeding in a similar manner as Section 2.1 for DEVS coupling, we state that STDEVS models can be coupled modularly, in such a way that an STDEVS coupled model N is defined by the structure:

$$N = (X_N, Y_N, D, \{M_d\}, \{I_d\}, \{Z_{i,d}\}, \ Select)$$

where the components are identical to those of the DEVS definition, except that the components M_d are now STDEVS structures.

The new introduced requisite for λ and ta functions about being measurable functions in atomic STDEVS models, will guarantee the desired measurability of $\mathcal{G}_{\text{int}}$ and $\mathcal{G}_{\text{ext}}$ in coupled STDEVS models. We will see this in more detail in Section 5.1.

5. Properties of STDEVS

This section studies the main properties of STDEVS. It first shows that STDEVS is closed under coupling (i.e. the coupling of atomic STDEVS models defines an equivalent atomic STDEVS model). Then, we redefine the concept of DEVS legitimacy in the context of stochastic DEVS.

5.1 Closure Under Coupling

We show that a coupled STDEVS model

$$N = \langle X_N, Y_N, D, \{M_d\}, \{I_d\}, \{Z_{i,d}\}, \ Select\rangle$$

with $M_d \in \{M_d\}$ being STDEVS atomic models for all d, defines an equivalent atomic STDEVS model, thus verifying STDEVS closure under coupling.

To achieve this, we find an atomic STDEVS model $M_{\text{ST}} = (X, Y, S_N, \mathcal{G}_{\text{int}_N}, \mathcal{G}_{\text{ext}_N}, P_{\text{int}_N}, P_{\text{ext}_N}, \lambda, ta)$ defined by the coupling expression N.

We begin by defining the relationships that are shared with the classic proof for deterministic DEVS [2]:

- $X = X_N, Y = Y_N$;
- $S_N = \underset{d \in D}{\times} \{(s_d, e_d)\}$ with $s_d \in S_d, e_d \in \Re$; each component of S_N has the form $s_N = (\ldots, (s_d, e_d), \ldots)$;
- $ta(s_N) = \min\{\sigma_d \mid d \in D\}$, with $\sigma_d = t_{a_d}(s_d) - e_d$;
- $d^* = Select(IMM(s_N))$, where IMM is the set of sub-models with minimum time to next event, i.e. $IMM = \{d | \sigma_d = ta(s_N)\}$;
- we have

$$\lambda(s_N) = \begin{cases} Z_{d^*,N}(\lambda_{d^*}(s_{d^*})) & \text{if } d^* \in I_N, \\ \oslash & \text{otherwise} \end{cases}$$

Then, we need to obtain the probability spaces that will represent the stochastic dynamics of the coupled model, as a result of the stochastic behavior of its atomic components.

First, for internal transitions, we define the set-collecting function:

$$\mathcal{G}_{\text{int}_N}(s_N) \triangleq \underset{d\in D}{\times}(\mathcal{G}_d \times \{\tilde{e}_d\})$$

where

$$\mathcal{G}_d = \begin{cases} \mathcal{G}_{\text{int}}(s_{d^*}) & \text{if } d = d^* \\ \mathcal{G}_{\text{ext}}(s_d, \hat{e}_d, x_d) & \text{if } x_d \neq \oslash \\ \{s_d\} & \text{otherwise} \end{cases}$$

with

$$x_d = \begin{cases} Z_{d^*,d}(\lambda_{d^*}(s_{d^*})) & \text{if } d^* \in I_d \\ \oslash & \text{otherwise} \end{cases}$$

$$\tilde{e}_d = \begin{cases} 0 & \text{if } d = d^* \text{ or } x_d \neq \oslash \\ \hat{e}_d & \text{otherwise} \end{cases}$$

and

$$\hat{e}_d = e_d + ta_{d^*}(s_{d^*}) - e_{d^*}.$$

Then, the sets $G_N \in \mathcal{G}_{\text{int}_N}(s_N)$ will have the form $G_N = (\ldots(G_d, \{e_d\}), \ldots)$ and will verify $G_N \subseteq S_N$.

We also call $\mathcal{F}_{\text{int}_N}(s_N) \triangleq \mathcal{M}(\mathcal{G}_{\text{int}_N}(s_N))$ the minimum sigma-field generated by $\mathcal{G}_{\text{int}_N}(s_N)$. Then, the probability measure for the internal transition process in N, $P_{\text{int}_N} : S_N \times 2^{S_N} \to [0, 1]$ is defined as:

$$P_{\text{int}_N}(s_N, G_N)$$

$$\triangleq P_{\text{int}_{d^*}}(s_{d^*}, G_{d^*}) \prod_{d|x_d\neq\oslash} P_{\text{ext}_d}(s_d, \hat{e}_d, x_d, G_d)$$

and it can be verified that the triplet $(S_N, \mathcal{F}_{\text{int}_N}(s_N), P_{\text{int}_N}(s_N, \cdot))$ is a probability space.

Similarly, for external transitions we define the set-collecting function:

$$\mathcal{G}_{\text{ext}_N}(s_N, e, x_N) \triangleq \underset{d\in D}{\times}(\mathcal{G}_d \times \{\tilde{e}_d\})$$

where

$$\mathcal{G}_d = \begin{cases} \mathcal{G}_{\text{ext}}(s_d, \hat{e}_d, x_d) & \text{if } x_d \neq \oslash \\ \{s_d\} & \text{otherwise} \end{cases}$$

$$\tilde{e}_d = \begin{cases} 0 & \text{if } x_d \neq \oslash \\ \hat{e}_d & \text{otherwise} \end{cases}$$

with

$$x_d = \begin{cases} Z_{N,d}(x_N) & \text{if } N \in I_d \\ \oslash & \text{otherwise} \end{cases}$$

and

$$\hat{e}_d = e_d + e.$$

The sets $G_N \in \mathcal{G}_{\text{ext}_N}(s_N, e, x_N)$ will also have the form $G_N = (\ldots(G_d, \{e_d\}), \ldots)$ and will verify $G_N \subseteq S_N$.

Again, we define $\mathcal{F}_{\text{ext}_N}(s_N, e, x_N) \triangleq \mathcal{M}(\mathcal{G}_{\text{ext}_N}(s_N, e, x_N))$ the minimum sigma-field generated by $\mathcal{G}_{\text{ext}_N}(s_N, e, x_N)$. Then, the probability measure for the external transition process in N, $P_{\text{ext}_N} : S_N \times \Re \times X \times 2^{S_N} \to [0, 1]$ is defined as

$$P_{\text{ext}_N}(s_N, e, x_N, G_N) = \prod_{d|x_d\neq\oslash} P_{\text{ext}_d}(s_d, \hat{e}_d, x_d, G_d)$$

and the triplet $(S_N, \mathcal{F}_{\text{ext}_N}(s_N, e, x_N), P_{\text{ext}_N}(s_N, e, x_N, \cdot))$ is a probability space.

Note that the functions $\lambda(s_N)$ and $ta(s_N)$ resulting from the coupling procedure are guaranteed to be measurable functions. It is given that any finite number of operations involving measurable functions over measurable sets always result in measurable functions and/or sets. The same argument can be applied to show that the subsets $G_N \in \mathcal{G}_{\text{int}_N}(\cdot)$ and $G_N \in \mathcal{G}_{\text{ext}_N}(\cdot)$ will be measurable sets.

This constructive proof shows that the generic coupled STDEVS model N is equivalent to the atomic STDEVS model M_{ST}. Thus, we can couple STDEVS models in a hierarchical way, encapsulating complex coupled models and coupling them with other atomic or coupled models. This property is just analogous to that of the classic deterministic DEVS formalism.

5.2 Legitimacy

Legitimacy in deterministic DEVS is a property that ensures that a model cannot perform an infinite number of transitions in a finite interval of time.

In STDEVS, this property must be redefined now expressing that the *probability* of having an infinite number of transitions in a finite interval of time is zero.

Given an STDEVS model, we consider a function $P_k : S \times \Re^+ \to [0, 1]$, so that $P_k(s, z)$ evaluates the probability that departing from state $s \in S$, after k internal transitions, the system accumulates an elapsed time equal or less than z.

We then define the *legitimacy* of STDEVS as follows.

Definition 3. *An STDEVS model is said to be* legitimate *when it verifies*

$$\lim_{k\to\infty} P_k(s, z) = 0, \quad \forall z < \infty, \ \forall s \in S. \tag{1}$$

In other words, an STDEVS model is legitimate when, starting from any state s, the probability of accumulating

a finite elapsed time after an infinite sequence of consecutive state changes is zero.

For instance, according to this definition, the model M_L introduced in Section 3.5 is legitimate, as it verifies (1).

Note that in this model it holds that

$$P_k(s,z) = P\left(\sum_{i=1}^{k} r_i < z\right), \quad \forall s \in S$$

where r_i are k uniformly distributed random numbers between zero and one. Given any small $\varepsilon > 0$, the weak law of large numbers establishes that

$$\lim_{k\to\infty} P\left(\left|\sum_{i=1}^{k} \frac{r_i}{k} - \frac{1}{2}\right| < \varepsilon\right) = 1$$

from where it can be easily derived that

$$\lim_{k\to\infty} P\left(\sum_{i=1}^{k} r_i > k\left(\frac{1}{2} - \varepsilon\right)\right) = 1$$

and then

$$\lim_{k\to\infty} P\left(\sum_{i=1}^{k} r_i < k\left(\frac{1}{2} - \varepsilon\right)\right) = 0.$$

Now, by taking $\varepsilon < 1/2$, the term $k(1/2 - \varepsilon)$ becomes greater than any z as k goes to ∞. Then,

$$\lim_{k\to\infty} P\left(\sum_{i=1}^{k} r_i < z\right) = 0$$

and thus the model M_L verifies the STDEVS legitimacy condition.

In order to complete Definition 3, we now derive a general expression for $P_k(s,z)$ in terms of the STDEVS characteristic functions P_{int} and ta. Note that the following construction of $P_k(s,z)$ is only an example among other valid procedures that might be found to check Definition 3.

We start by selecting a small positive scalar ε, with $0 < \varepsilon \ll 1$ and defining the sets $S_{x,\varepsilon}$ composed by states verifying a bounding criteria for their time advance values, according to

$$S_{x,\varepsilon} \triangleq \{s \mid x \le ta(s) < x + \varepsilon\}. \tag{2}$$

Now we introduce the function $p_k(s,z,\varepsilon)$ that evaluates the probability that the system, departing from state s, accumulates an elapsed time in the interval $[z, z+\varepsilon)$ after k internal transitions. For $k = 1$, and taking into account the function P_{int} and the sets $S_{x,\varepsilon}$ defined above, it follows that

$$p_1(s,z,\varepsilon) \triangleq P_{\text{int}}(s, S_{z,\varepsilon}). \tag{3}$$

Let us suppose that we know the expression of p_k for certain values of k. In order to find the expression for p_{k+1}, we first define $R_k(z,x,c,\varepsilon)$ as the set of the states $s \in S_{x,\varepsilon}$ with probability in the interval $[c, c+\varepsilon)$ of accumulating an elapsed time between z and $z+\varepsilon$ after k transitions. Formally,

$$R_k(z,x,c,\varepsilon) \triangleq \{s \in S_{x,\varepsilon} \mid c \le p_k(s,x,z) < c+\varepsilon\}. \tag{4}$$

Then, assuming that ε is small enough, we can evaluate $p_{k+1}(s,z,\varepsilon)$ as the probability of making a step of duration between x and $x+\varepsilon$ followed by k steps with a total duration between $z-x$ and $z-x+\varepsilon$. This probability can be calculated as the sum of the probabilities of passing through the different disjoint sets $R_k(z-x,x,c,\varepsilon)$ with different values of c (between zero and one) and x (between zero and z). This is,

$$p_{k+1}(s,z,\varepsilon) = \sum_{m_x=0}^{[z/\varepsilon]} \sum_{m_c=0}^{[1/\varepsilon]} P_{\text{int}}(s, R_k(z-\tilde{x},\tilde{x},\tilde{c},\varepsilon))\tilde{c} \tag{5}$$

with

$$\tilde{x} = m_x\varepsilon, \quad \tilde{c} = m_c\varepsilon. \tag{6}$$

We have found a general expression for p_k with $k \ge 1$. Then, we can build $P_k(s,z)$ according to

$$P_k(s,z) = \lim_{\varepsilon\to 0} \sum_{m_x=0}^{z/\varepsilon} p_k(s,\tilde{x},\varepsilon) \tag{7}$$

in cases where this limit exists[4]. This expression, together with (1), expresses a legitimacy condition for STDEVS as a function of $P_{\text{int}}(s,\cdot)$ and $ta(s)$.

6. DEVS, RND Functions and STDEVS

In this section we study the links between deterministic DEVS and stochastic STDEVS. We first show that we can build STDEVS models using DEVS conventional models equipped with RND functions in the transition functions. Then, we also show that DEVS models with measurable functions are particular cases of STDEVS and that this property allows us to formally couple together DEVS and STDEVS models.

6.1 DEVS Models with Functions RND

We show that a DEVS-like model whose transition functions depend on random variables (typically generated using RND functions) define, under certain conditions, an STDEVS model.

4. Otherwise, any other valid procedure to find an expression for $P_k(s,z)$ according to Definition 3 can be used.

Thus, it will first be clear that STDEVS can represent any *practical* and *consistent* stochastic DEVS model defined by the usual method of using RND functions. Second, this property allows us to define and simulate STDEVS models in a very simple and straightforward way, getting rid of the need to use probability spaces which add complexity to the model definition structure and terminology.

As before, we call DEVS models whose transition functions depend on RND functions DEVS-RND models. We distinguish the concept of DEVS-RND (a definition for mathematical modeling purposes) from the concept of Zeigler's original utilization of a pseudo-random number generator in a DEVS transition function [1] (a specification for practical simulation purposes on a computer). Eventually (but not necessarily) a given DEVS-RND model specification can be practically simulated using pseudo-random sequences in DEVS transition functions with satisfactory results.

Also, we say that a DEVS-RND model is measurable when all of its functions $(\delta_{\text{int}}, \delta_{\text{ext}}, ta, \lambda)$ are measurable on the corresponding sets.

Theorem 1. *A measurable DEVS-RND model*

$$M_D = (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta)$$

in which its state change functions δ_{int} and δ_{ext} depend dynamically on a random experiment through a random variable r (i.e. $\delta_{\text{int}} = \delta_{\text{int}}(s, r)$ and $\delta_{\text{ext}} = \delta_{\text{ext}}(s, e, x, r)$) with $r \in R \subseteq \Re^n$ characterized by a probability measure $P(r \in B \mid B \in \mathcal{B} \subseteq 2^R)$, defines an equivalent STDEVS model[5].

Proof. We obtain an STDEVS model

$$M_{\text{ST}} = (X, Y, S, \mathcal{G}_{\text{int}}, \mathcal{G}_{\text{ext}}, P_{\text{int}}, P_{\text{ext}}, \lambda, ta)$$

equivalent to M_D, assuming that X, Y, S, λ, ta are identical for M_D and M_{ST}. Thus, we only need to find $\mathcal{G}_{\text{int}}, \mathcal{G}_{\text{ext}}, P_{\text{int}}$ and P_{ext}.

We start by defining the collecting set $\mathcal{G}_{\text{int}}(s)$ in relation to the sigma-field $\mathcal{B}$ of the random experiment. For each set $B \in \mathcal{B}$ and for each state $s \in S$, we define the *preimage set* $G_{s,B} \subseteq S$ according to

$$\hat{s} \in G_{s,B} \iff \exists\, r \in B / \delta_{\text{int}}(s, r) = \hat{s}.$$

Since δ_{int} is a measurable function and the set B is measurable, then as a result the set $G_{s,B}$ is also measurable. Then, we define $\mathcal{G}_{\text{int}}(s)$ as

$$\mathcal{G}_{\text{int}}(s) \triangleq \{G_{s,B} | B \in \mathcal{B}\}.$$

5. We use $\mathcal{B}$ to denote the sigma-field where the function P is defined.

Therefore, for the system being in state s, the probability of transition to a new state belonging to $G_{s,B} \in \mathcal{G}_{\text{int}}(s)$ is

$$P_{\text{int}}(s, G_{s,B}) = P(r \in B).$$

Then, for each state $s \in S$, the function $P_{\text{int}}(s, \cdot)$ is a probability measure in the measurable space $(S, \mathcal{F}_{\text{int}}(s))$, being $\mathcal{F}_{\text{int}}(s) = M(\mathcal{G}(s))$ the minimum sigma-field generated by $\mathcal{G}_{\text{int}}(s)$. This is proved by verification of the following axioms:

1. $P_{\text{int}}(s, G_{s,B}) \geq 0$ because $P_{\text{int}}(s, G_{s,B}) = P(r \in B) \geq 0$;
2. $P_{\text{int}}(s, S) = 1$, given $\int (s, r) \in S,\ \forall s, r$;
3. let $B_1, B_2 \in \mathcal{B}$, then, if $G_{s,B_1} \cap G_{s,B_2} = \oslash \Rightarrow B_1 \cap B_2 = \oslash$, therefore, the following holds

$$\begin{aligned} P_{\text{int}}(s, G_{s,B_1} \cup G_{s,B_2}) &= P(r \in B_1 \cup B_2) \\ &= P(r \in B_1) + P(r \in B_2) \\ &= P_{\text{int}}(s, G_{s,B_1}) \\ &+ P_{\text{int}}(s, G_{s,B_1}). \end{aligned}$$

So far, we obtained $\mathcal{G}_{\text{int}}$ and P_{int} for the STDEVS model M_{ST} departing from the DEVS-RND model M_D definition and the randomness condition incorporated in $\delta_{\text{int}}(s, r)$.

In the case of $\mathcal{G}_{\text{ext}}$ and P_{ext} we proceed analogously, this time replacing the state s by the triplet (s, e, x) for the analysis. This concludes the proof. ■

Consider now the particular case $r \in R = [0, 1]^n \subset \Re^n$ with uniform distribution. We say that r is uniformly distributed when every component of r have uniform distribution over the interval [0, 1]:

$$r_i \sim U(0, 1), \quad i = 1, 2, \ldots, n$$

This is the typical case emulated by *pseudo-random sequence generators* used in most of the programming languages (we call them *RND*). It is interesting to take a look separately for this particular case given that STDEVS models are usually simulated using *RND* functions.

The following is then a corollary of Theorem 1, particularizing the properties of STDEVS models when using *RND* functions within the transition definitions.

Corollary 1. *A measurable DEVS-RND model in which $\delta_{\text{int}}(s, r)$ depends on n functions RND (i.e. $r \sim U(0, 1)^n$) defines an STDEVS equivalent model.*

This corollary does not need any proof, given that it is a particular case of Theorem 1, taking $R = [0, 1]^n$. Anyway, we can make explicit reference of the components of the resulting STDEVS model.

Proceeding as in the general case, for each *image set* $G_{s,B} \in \mathcal{G}(s)$, the probability of transitioning from state s to a new state belonging to the set $G_{s,B}$ will be

$$P_{\text{int}}(s, G_{s,B}) = P(r \in B)$$

which turns out to be the Lebesgue measure for the set B.

6.2 DEVS and STDEVS

In the case that one (or both) transition function(s) is deterministic, it can still be defined as $\delta(\cdot, r)$, but in such a way that it is independent on r. Hence, the whole previous analysis remains valid.

Following this reasoning, the theorem presented here is an alternative way of proving that deterministic measurable[6] DEVS is a particular case of stochastic STDEVS, where randomness is removed from state transition dynamics.

Finally, if we consider that both transition functions are deterministic (the DEVS case) and use the same concept of defining them with $\delta(\cdot, r)$ independent on r, the following corollary can be derived.

Corollary 2. *A deterministic and measurable DEVS model always defines an equivalent stochastic STDEVS model.*

6.3 Coupling of DEVS and STDEVS

An important feature of STDEVS models should be its transparent integration with measurable DEVS models into hybrid structures. The coupling of DEVS atomic models (along with their closure under the coupling property) was defined in [2], and has also now been defined for STDEVS models in Sections 4.3 and 5.1. Now, in order to guarantee connectivity between both types of models it is sufficient to prove that a single measurable DEVS atomic model structure can always be connected to a single STDEVS atomic model structure. To this aim we can make use of Corollary 2, and consider any given measurable DEVS atomic model as its STDEVS equivalent. Thus, the closure under coupling for STDEVS guarantees that a valid connection between both types of models results in a new STDEVS coupled model.

This also allows for the composition of models by the hierarchical connection of measurable DEVS models and STDEVS models. This implies that a coupled STDEVS can be used, in turn, into a more complex coupled model; a procedure called hierarchical coupling. The procedure can be extended as many times as needed and at any hierarchical level, thus allowing us to naturally combine measurable DEVS and STDEVS models according to the needs of the user.

6. As in the case of DEVS-RND, we say that a DEVS model is measurable when all of its functions are measurable.

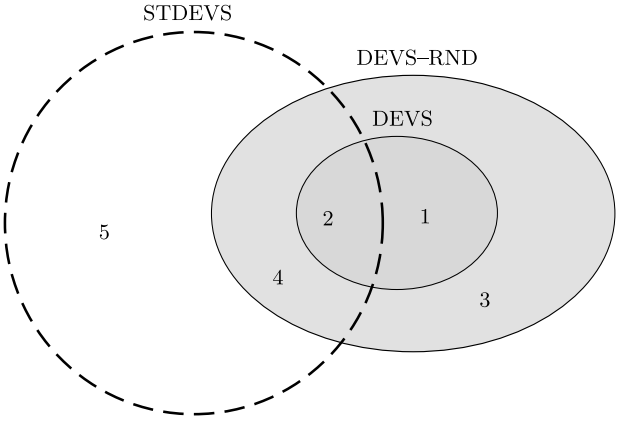

Figure 2. Relationship between categories of models.

It is worth noting that it is enough to have one single STDEVS atomic model as part of a more complex model structure, in order to need an STDEVS description for the whole coupled system.

6.4 Summary and Discussion

We close the theoretical body of our work with a classification of the formalisms discussed so far, namely DEVS, DEVS-RND and STDEVS, from the point of view of the main properties that distinguish the models belonging to them, i.e. measurability and dimensionality of their sets and functions. Also, we map to this classification those initial problems found in the motivational models introduced in Section 3 to provide a comprehensive understanding of the context in which they operate.

Each set in the Venn diagram of Figure 2 represents a particular family of models. The limits between the sets imply some kind of distinction according the different abilities of each model family to represent certain modeling problems. The formalisms are conformed as DEVS $= 1 \cup 2$, DEVS-RND $=$ DEVS $\cup\, 3 \cup 4$ and STDEVS $= 2 \cup 4 \cup 5$.

The difference between models belonging to $1 \cup 2$ (classic DEVS) and those belonging to $3 \cup 4$ is that the latter are able to represent stochastic behavior (to a limited extent). As we saw in Section 3.1, the functions in classic DEVS cannot formally represent stochastic behavior, because this leads to a situation where they lose the property of assigning a unique value to each element of a given input set. As depicted in Figure 2, all DEVS models define DEVS-RND models.

Another important difference is between models belonging to $2 \cup 4$ and those belonging to $1 \cup 3$. Models in $2 \cup 4$ are required to operate over *measurable* sets, while models in $1 \cup 3$ are not. This is why the models in $2 \cup 4$ are called *measurable DEVS-RND*, and the models in 2 are called *measurable DEVS*.

STDEVS was defined by taking into account that consistent stochastic processes must operate over measurable sets. In that way, STDEVS cannot represent models in $1 \cup 3$. However, STDEVS extends the domain of DEVS-RND. The family of models number 5 is exclusive to the STDEVS representation. These models are different from the rest because of their ability to represent stochastic behavior over (measurable) *infinite-dimensional* spaces, which provide the *generality* property we asked for in the motivational example of Section 3.2. STDEVS also defines measurable DEVS-RND models (family 4) and measurable DEVS models (family 2). The inability of STDEVS to represent models in $1 \cup 3$ is in fact an advantage, because it prevents STDEVS from defining inconsistent models (both at the atomic and coupled levels).

All of the aforementioned remarks can be illustrated using the motivating examples discussed previously in Section 3. Model M_U in Section 3.1 belongs to family 4 (i.e. it is a DEVS-RND model operating over measurable sets). It also defines an STDEVS model. The model described in Section 3.2 belongs clearly to family 5 (i.e. it is an STDEVS model operating over infinite-dimensional spaces that cannot be represented by DEVS-RND). Model M_V in Section 3.3 is inconsistent because it attempts to assign a measure over a non-measurable set, thus, it is a DEVS-RND model in family 3, but it does not define an STDEVS model. In Section 3.4 the model M_A is a deterministic DEVS model in family 1 that does not define an STDEVS model, and M_U is a consistent DEVS-RND model of family 4 that does define an STDEVS model. While M_A is not inconsistent itself, when coupled with M_U the result is an inconsistent model that behaves in the same way as model M_V in family 3 and does not define an STDEVS model.

One final concept about this categorization of DEVS-related models into families is, again, the distinction between modeling and simulation. Most practical models ever implemented in DEVS-based simulators belong to the zone $2 \cup 4$. This is mainly due to the finiteness of the data-representation capabilities of known computing platforms, which inherently lead to a *finite and measurable* representation of perhaps more sophisticated theoretical models for *simulation* purposes. Moreover, in most practical cases, after a system is *modeled* mathematically as a measurable DEVS-RND model (zone 4 in Figure 2), it still needs to be assigned a computationally implementable algorithm that approximates satisfactorily some selected statistical properties. When such an algorithm exists, and is selected, the measurable DEVS-RND model is ultimately *simulated* as a deterministic DEVS model (zone 2 in Figure 2) that uses pseudo-random number sequences (i.e. deterministic sequences, strictly speaking) to evaluate state transitions.

However, as we showed throughout the present work, very important issues arise when dealing with the theoretical stochastic *modeling* capabilities and limitations of the formalisms, not limiting the analysis to the computationally practical subsets of possibilities. These issues become evident and necessary when introducing the probability spaces theory into the DEVS framework.

7. Case Study

In this section we develop two case studies to illustrate the use of STDEVS for modeling stochastic systems and its relation with the further simulation process. We first introduce a model of a task load-balancing system. We then present the model of a hybrid networked control system. Using the theory presented we see that the practical measurable DEVS-RND representations of the random processes are consistent with their STDEVS specification in terms of probability spaces.

7.1 A Foreword on the Process of Stochastic Modeling and Simulation with STDEVS

As discussed throughout this work (and stressed in particular in Section 6.4) the distinction between modeling and simulation becomes a sensitive issue when dealing with stochastic systems.

Indeed, we already know that there exist practical simulatable DEVS models using RND functions that can lead to completely incorrect probabilistic results (e.g. those involving unmeasurable sets, that can be defined with DEVS); and there also exist completely valid stochastic mathematical models that can never be simulated with today's digital computers (e.g. those involving infinite-dimensional spaces, that can be defined with STDEVS).

We also already know that STDEVS is a formal framework that allows for consistent and general mathematical *modeling* of DEVS stochastic systems, relying on a sound theory supporting them: the probability spaces theory.

Now, when it comes to the subject of practical case studies using STDEVS, it is important to focus on the stochastic *model-to-simulation process* that leads from a formally sound stochastic model to an executable piece of code. A diagram of this process involving STDEVS is shown in Figure 3. The full process comprises three activities and two transitional steps.

Activities represent the action of obtaining a DEVS-based representation of a given stochastic system. The process can start directly at any *Activity*. When the input information for an Activity is only the real-world stochastic system to be modeled, no previous transitional *Step* is involved. When an Activity starts from a previous completed Activity, the transitional Step represents the adaptation of the previous state of the stochastic model to a new state of representation (i.e. a model transformation procedure).

We claim that *by starting the stochastic modeling process at any point other than Activity* (*a*) *increases the methodological risk of obtaining an inaccurate model*. Nevertheless, in some cases the risk can be sufficiently

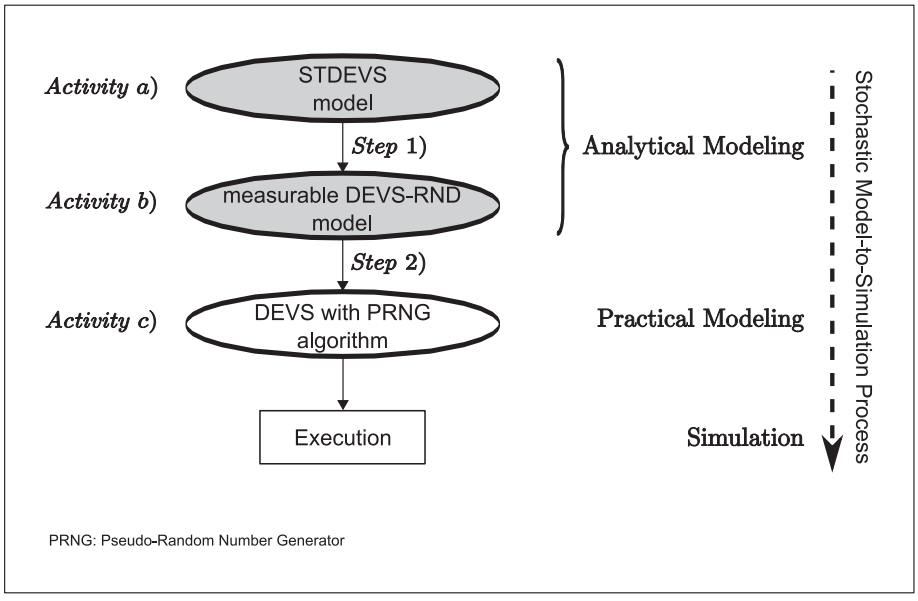

Figure 3. Stochastic model-to-simulation process. Activities and transitional steps.

understood and considered low enough to proceed with the process by starting at any desired point.

For example, starting directly in *Activity* (*c*) may or may not lead to the reproduction of a valid and consistent stochastic model, and its accuracy relies on the theoretical background and experience of the programmer. Of course, well-known and widely used practical probabilistic distributions are readily available for their code implementation by means their algorithmic descriptions [20] or even reusable code libraries of public domain [21].

Nevertheless, it is obvious that these practical shortcuts (when correct) are confined to represent a very particular subset of stochastic models: the subset of those models that (*a*) are representable by measurable DEVS-RND descriptions and (*b*) preserve satisfactorily some stochastic properties when implemented in a DEVS simulator with pseudo-random number generators.

Thus, the existence of the above-mentioned wide family of practical stochastic algorithms does not provide any generalization of their mathematical description, and is not enough to substitute a formal and general stochastic modeling formalism. On the other hand, STDEVS (while closer to the analytic description of the models than to their practical implementation) provides the mathematical tool to assess the correctness of any possible stochastic DEVS model in the context of probability spaces theory, including those frequently implemented by DEVS practitioners.

Put another way, every possible *measurable DEVS* or *measurable DEVS-RND* simulatable model running on a digital computer, is a practical implementation of an STDEVS model. At this point, it is clear that by simply throwing pseudo-random number generators into the transition functions of our favorite DEVS simulator, we cannot claim for guarantees of the measurability property of their equivalent DEVS-RND models; afterwards, we cannot ensure their representation as an STDEVS model, and consequently we cannot make any assumption about their consistency with the probability spaces theory.

Thus, it makes sense to work in the opposite way, as suggested in Figure 3. This is, from the STDEVS model specification, down to the DEVS-RND model specification, and finally down to the implementation on a digital computer (i.e. DEVS with pseudo-random generators). If we succeed through these steps, the model-to-simulation process itself guarantees an executable code that complies with the theoretical requirements needed to represent a proper stochastic model.

Note: As we shall see, in practice, things become much easier when we know in advance that we are modeling some classic probabilistic distributions, that are known to operate over well-defined probability spaces. In situations such as these, we can make use of Theorem 1 and claim that the *measurable DEVS-RND* descriptions of those models *are also STDEVS models*. This situation allows us to skip *Activity* (*a*) and *Step* (1) in the modeling process[7], which is an advantage in terms of the reduction of modeling effort. Then, from *Activity* (*b*) downwards in

7. With this decision, we accepted the risk involved because (a) we know that the risk exists and (b) we have a sufficient amount of previous knowledge about the problem at hand to be confident that the risk of being wrong is extremely low.

Figure 3, the process does not differ from the typical procedure being carried out through the years for DEVS modeling and simulation of stochastic systems.

Finally, the aim of the examples below is to show the complete stochastic model-to-simulation process, involving STDEVS explicitly in some situations, and resorting to Theorem 1 in other cases, completing the usual DEVS modeling and simulation procedures with subsequent results analysis. As we want to reach successfully simulatable situations, we pick examples that do not pose any difficulty in completing the full process.

7.2 Load Balancer Model

The example *load-balancing model* (LBM) introduced in this section is a simplification of a computing system that processes successive tasks. This simple example shows a system in which the dynamics fully depend on random experiments.

The LBM is formed by the following atomic models: *load generator* (LG), *weighted balancer* (WB) and two *servers* (S1,S2) with no queuing policy (i.e. the tasks arriving at a busy server are discarded). The set {WB,S1,S2} forms the subsystem *cluster* (CL), a coupled model.

As before, transition functions are expressed in terms of $r \sim U(0,1)$, namely $\delta_{\text{int}}(\cdot) = \delta_{\text{int}}(s,r)$ and $\delta_{\text{ext}}(\cdot) = \delta_{\text{ext}}(s,e,x,r)$.

7.2.1 Load Generator (LG)

This model generates a number of tasks per time unit according to a discrete Poisson random distribution being d_r the *mean expected departure rate*. It can be proven that the inter-departure time σ_k between tasks k and $k+1$ is exponentially distributed according to $P(\sigma_k \leq t) = 1 - e^{-at}$, where $a = d_r$ and $1/a$ is the mean expected value. We assume that the model LG generates only one type of task ($task_1$) which goes out through the only output port (out_1). The LG model does not have any inputs, thus only internal transitions are possible. The STDEVS definition for LG is

$$M_{\text{ST}}^{\text{LG}} = (X, Y, S, \mathcal{G}_{\text{int}}, \mathcal{G}_{\text{ext}}, P_{\text{int}}, P_{\text{ext}}, \lambda, ta)$$

with deterministic components:

- $X = \oslash$, $Y = \{(task_1, out_1)\}$;
- $S = \Re_0^+$;
- $\lambda(s) = (task_1, out_1)$;
- $ta(s) = s$;

and stochastic functions:

- $\mathcal{G}_{\text{int}}(s) = \{A_t \mid t \geq 0\}$, $A_t = [0, t)$;
- $P_{\text{int}}(s, G) = P_{\text{int}}(s, A_t) = 1 - e^{-at}$, $G \in \mathcal{G}_{\text{int}}$.

As we can see the stochastic description for the inter-departure time of tasks is mapped directly to the function P_{int} through the corresponding cumulative distribution function. As only internal transitions are possible, we do not need to define $\mathcal{G}_{\text{ext}}$, P_{ext}.

To implement this STDEVS model in a digital computer, the probabilistic description must be translated into an algorithm to be evaluated into the internal transition code, representing the associated DEVS $\delta_{\text{int}}(\cdot)$ function. It is, we have to transition from *Activity* (a) to *Activity* (b) completing *Step* (2) in the process of Figure 3. To accomplish this, and according to our previous definitions, we define

$$\delta_{\text{int}}(s, r) = -(1/a)log(r)$$

where by means of the inverse transformation method we have obtained an exponentially distributed function making use of a uniform distributed variable $r \sim U(0,1)$ available as a RND() function in most programming languages.

Consequently, the equivalent measurable DEVS-RND specification for LG is

$$M_D^{\text{LG}} = (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta)$$

where:

- $X = \oslash$;
- $Y = \{(task_1, out_1)\}$;
- $S = \Re_0^+$;
- $\delta_{\text{int}}(s, r) = -(1/a)\log(r)$;
- $\delta_{\text{ext}}(s, e, x, r) = s$;
- $\lambda(s) = (task_1, out_1)$;
- $ta(s) = s$.

In this component, the next randomly calculated inter-departure time is stored in the real-valued state s, which is then used by the time advance function $ta(s) = s$ making LG 'inactive' during that period[8].

Note that by making state s an arbitrarily designed n-tuple we are able to apply independent random distributions to decide the next state value for each of the n elements in s after a state transition. Thus, by simply including the time advance σ as one of the n component elements of s and afterwards using σ to evaluate $ta(s)$, we are readily able to apply any stochastic distribution to the lifetime of the next state. If m of the n components of s ($m \leq n$) are to be decided with independent random generators, then we can use a random vector $r \sim U(0,1)^m$ of dimension m.

8. Similar reasoning can be applied for the rest of the components, where the state values are used for storage purposes.

Note: From now on, we build the models of this example directly in DEVS-RND. We make use of Theorem 1 and only refer to the DEVS-RND form of those components with some form of stochastic behavior, containing RND() functions in the algorithms that evaluate transitions. We can do this because the models we create represent classic probability distributions belonging to well-defined finite-dimensional probability spaces, thus yielding measurable DEVS-RND models. This is an easier modeling approach than defining the STDEVS structure of the model (as we did before for LG), because in DEVS-RND we do not need to define the stochastic functions $\mathcal{G}_{\text{int}}$, $\mathcal{G}_{\text{ext}}$, P_{int}, P_{ext}. Still, by means of Theorem 1, the corresponding equivalent STDEVS model can always be obtained from a *measurable* DEVS-RND model, building the STDEVS structure following the same reasoning used for LG.

7.2.2 Weighted Balancer (WB)

The WB component delivers the incoming tasks arriving at input port (Port: inp_1) to the output ports out_1 and out_2 based on a balancing factor $b_f \in [0, 1]$ that determines the weight relation between both ports. For $b_f = 0.5$ both outputs have the same weight and therefore the outgoing load will be balanced equiprobably. For $b_f > 0.5$, out_1 is privileged and for $b_f < 0.5$, out_2 is privileged, in a linear fashion. The tasks accepted belong to a set $T = \{task_1, \ldots, task_m\}$ with m different possible tasks.

The measurable DEVS-RND definition M_D^{WB} for WB is

$$M_D^{\text{WB}} = (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta)$$

with:

- $X = T \times \{inp_1\}$, $Y = T \times \{out_1, out_2\}$;
- $S = T \times \{out_1, out_2\} \times \Re_0^+$;
- $\lambda(w, p, \sigma) = (w, p)$;
- $ta(w, p, \sigma) = \sigma$.

The state is a triplet $s = (w, p, \sigma)$, where w represents the last task received, p is the port where that task is delivered and σ is the time advance. For our example $T = \{task_1\}$. After receiving an event (x_v, x_p) the new state must be evaluated by

$$\delta_{\text{ext}}((w, p, \sigma), e, (x_v, x_p), r) = (x_v, \tilde{p}, 0)$$

with

$$\tilde{p} = \begin{cases} out_1 & \text{if } r < b_f \\ out_2 & \text{otherwise} \end{cases}$$

Finally, the internal transition will be

$$\delta_{\text{int}}((w, p, \sigma), r) = (w, p, \infty)$$

in this case, independent of r.

7.2.3 Server 1 and Server 2 (S1,S2)

The servers S1 and S2 are components that receive the tasks delivered by the balancer WB. The servers process each task received, which takes a service time s_t. Once processed, the task is sent out to a sink, where it is recognized as a completed task. The service time variable s_t is distributed exponentially with $P(s_t \leq t) = 1 - e^{-bt}$, and its mean expected value is $1/b$.

There is no queuing policy nor preemption defined for the servers. If a new task arrives at a busy server, the task is ignored.

We give the measurable DEVS-RND definition $M_D^{S_n}$ with $n = 1, 2$ for S1 and S2, respectively:

$$M_D^{S_n} = (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta)$$

where:

- $X = T \times \{inp_1\}$, $Y = T \times \{out_1\}$;
- $S = T \times \{false, true\} \times \Re_0^+$;
- $\lambda(w, busy, \sigma) = (w)$;
- $ta(w, busy, \sigma) = \sigma$.

The state is a triplet $s = (w, busy, \sigma)$, where w represents the last task received, *busy* represents the status of the server (if *busy* = *true* the server is processing a task and if *busy* = *false* the server is free) and σ is the time to the next scheduled event. For our example, we have $T = \{task_1\}$ and only one input port and one output port. After receiving an event (x_v, x_p) the new state will be evaluated according to:

$$\delta_{\text{ext}}((w, busy, \sigma), e, (x_v, x_p), r) = (\tilde{w}, true, \tilde{\sigma})$$

with

$$\begin{cases} \tilde{w} = x_v, \tilde{\sigma} = -(1/b)\log(r) & \text{if } not(busy) \\ \tilde{w} = w, \tilde{\sigma} = \sigma - e & \text{if } busy \end{cases}$$

with $r \sim U(0, 1)$. The internal transition will be completed as

$$\delta_{\text{int}}((w, busy, \sigma), r) = (w, false, \infty)$$

independently of r.

7.2.4 LBM Coupled Model

As discussed earlier, this model is intended to show a scenario where random variables affect all of its building components. Here, we have a Poisson process dominating the task generation, a uniform process (with a latter deterministic bias) affecting the balancing between two servers

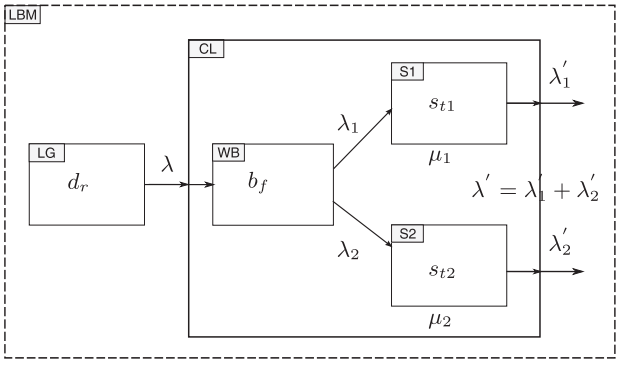

Figure 4. Topology of the LBM example.

and a negative exponential process representing task servicing times at servers. Nevertheless, the implementation always rely on the use of a uniform distributed variable $r \sim U(0, 1)$.

In Figure 4 the model topology is represented along with the main model parameters and derived traffic magnitudes that will be used in the next section.

7.2.5 Simulation Results

Based on the measurable DEVS-RND specification of the components and their interconnections, we built the complete model in a DEVS simulation tool (PowerDEVS [10]). This implies completing the transitional *Step* (2) and the *Activity* (*c*) in the process depicted in Figure 3. We do not go in further detail about this part of the process because it does not introduce any new concepts and does not present interesting complexities. Also, we refer to the model as the *STDEVS model*, given that we already know that what is obtained following the proposed stochastic model-to-simulation process are all particular instantiations explained by the general STDEVS framework.

Then, we ran several simulations at different system operating points. In order to validate the STDEVS model, we used the simulation results and some formulas known in queuing theory [22]. We then compared the expected theoretical values against the simulated results.

A single server with no queuing capacity can be described by an $M/M/m/m$ system with $m = 1$. This description assumes exponential inter-arrival times and exponential service times (which match our case). For the ith server we have the parameters λ_i (*arrival rate*) and μ_i (*service rate*). The *traffic intensity* is defined as

$$\rho_i = \lambda_i/\mu_i. \tag{8}$$

Owing to the limited buffering capacity (in our simplest case, only the servicing task can be 'buffered') there is a probability of losing tasks, which will never be serviced. This probability is denoted by P_{loss_i} (*probability of loss*) and is related with the traffic intensity by *Erlang's loss formula* [22] in its simplest form for a single server:

$$P_{\text{loss}_i} = \rho_i/(1 + \rho_i). \tag{9}$$

The ith server will see at its input port an *effective arrival rate* of

$$\lambda_i' = \lambda_i(1 - P_{\text{loss}_i}) \tag{10}$$

which under stability conditions[9] is equal to the *server throughput* at its output port. In our LBM example, we have $i = 1, 2$ for the two servers in the cluster (CL) submodel. The *total system throughput* λ' must be $\lambda' = \lambda_1' + \lambda_2'$, hence being a function of the *total system arrival rate* λ and the traffic intensities ρ_1, ρ_2 at the servers.

These magnitudes are calculated from the model parameters set up for simulation: d_r (mean departure rate at LG, in tasks per second), b_f (balancing factor at WB), s_{t1}, s_{t2} (mean service time at S1 and S2 respectively, in seconds) as follows:

$$\begin{aligned} \lambda &= d_r \\ \mu_1 &= 1/s_{t1}, \quad \lambda_1 = b_f\lambda \\ \mu_2 &= 1/s_{t2}, \quad \lambda_2 = (1 - b_f)\lambda. \end{aligned} \tag{11}$$

Now, with (8) and (11) in (9) we derive the internal loss probabilities

$$\begin{aligned} P_{\text{loss}_1} &= \frac{b_f d_r s_{t1}}{1 + b_f d_r s_{t1}}, \\ P_{\text{loss}_2} &= \frac{(1 - b_f) d_r s_{t2}}{1 + (1 - b_f) d_r s_{t2}}. \end{aligned} \tag{12}$$

Finally, we want to express the *total system throughput* in terms of a *total system loss probability* P_{loss} as we did for the individual servers. So with (10) and (12) we obtain

$$\begin{aligned} P_{\text{loss}} &= b_f P_{\text{loss}_1} + (1 - b_f) P_{\text{loss}_2} \\ \lambda' &= \lambda(1 - P_{\text{loss}}). \end{aligned} \tag{13}$$

With (13) we completely characterize the system in terms of offered load, loss probabilities and effective throughput. Figure 5 shows the theoretical curves for P_{loss}, P_{loss_1}, P_{loss_2} and λ' as functions of b_f in a *test scenario 1* chosen as $TS_1 = \{d_r = 10,\ b_f \in [0, 1],\ s_{t1} = 0.2,\ s_{t2} = 0.2\}$. In the same figure we have plotted simulation results for the STDEVS model LBM parameterized according the scenario TS_1, at a set of illustrative operational points sweeping b_f between zero and one.

9. In lossy systems, the *effective traffic intensity* $\rho_i' = \lambda_i'/\mu_i$ is always $\rho_i' < 1$ so the typical stability condition $\lambda_i/\mu_i < 1$ is not required. Finite buffer systems are always stable since arriving tasks are lost when the number of tasks in the system exceeds system capacity.

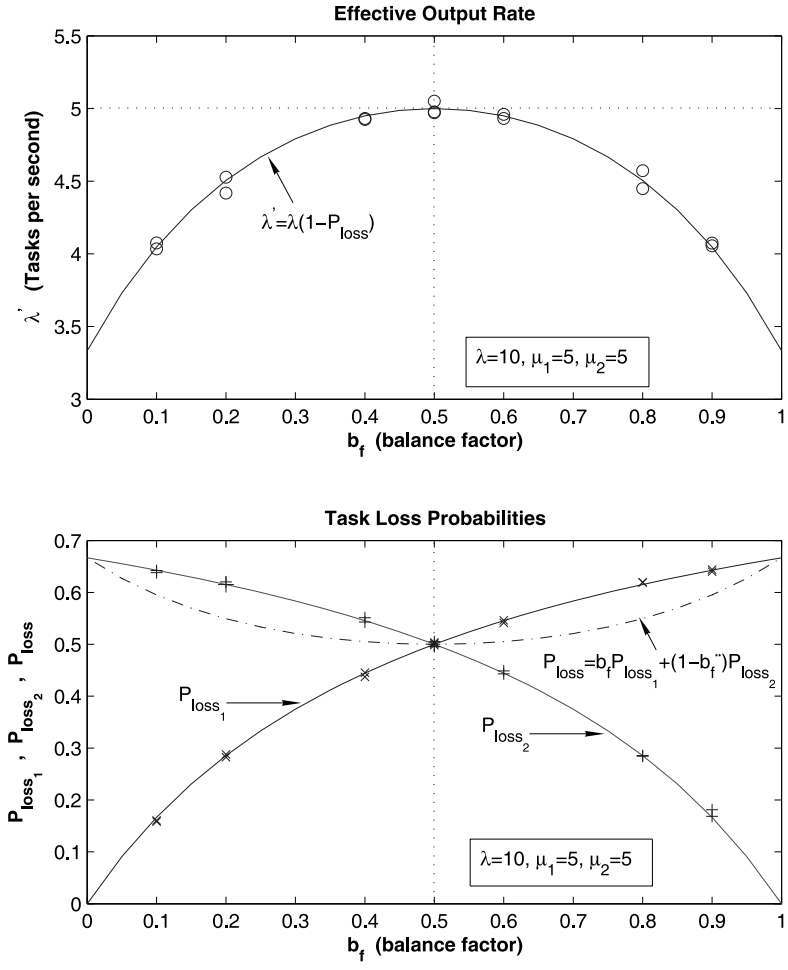

Figure 5. Simulation results (points) versus theoretical curves (lines). Test scenario 1: $d_r = 10$, $b_f \in [0, 1]$, $s_{t1} = 0.2$, $s_{t2} = 0.2$.

It can be observed that the simulation results closely match the expected theoretical curves, for 15 successive repetitions at each point.

The simulation point values were derived from the output event log files produced by the simulation runs, using the computed *task rate*[10] variables, thus obtaining λ'^{sim} and $P^{\text{sim}}_{\text{loss}_i} = 1 - (\lambda'^{\text{sim}}_i / \lambda^{\text{sim}}_i)$. The statistical properties of the random variables produced by the atomic models were verified to match with those expected: uniform distribution for b_f, discrete Poisson distribution for λ and exponential distribution for s_{t1} and s_{t2}. This also produced Poisson distributed series of values for all of the observed task rates, as expected.

10. A general λ^{sim}_k task rate at an arbitrary observation place k is: $\lambda^{\text{sim}}_k = NumberOfTasksLogged_k / TotalSimulationTime$.

7.3 Networked Control System

In this example we show another practical DEVS representation of random processes, consistent with their STDEVS specifications, in a system including hybrid modeling and control theory. We present the hybrid model of a Networked Control System (NCS) [23, 24] with components driven by continuous time and discrete time signals. NCSs are control systems with control loops closed through real-time networks, where the information of the main signals is distributed throughout system components over the underlying networks. The conceptual block diagram of the model is shown in Figure 6. The system consists in a LTI (linear time-invariant) plant modeled as a SISO (single-input single-output) system, in which the output signal $y'(t)$ must be kept in a certain reference input value. To accomplish this, a feedback control system is implemented, which utilizes a shared digital control network for communicating the sensed output data to the control module.

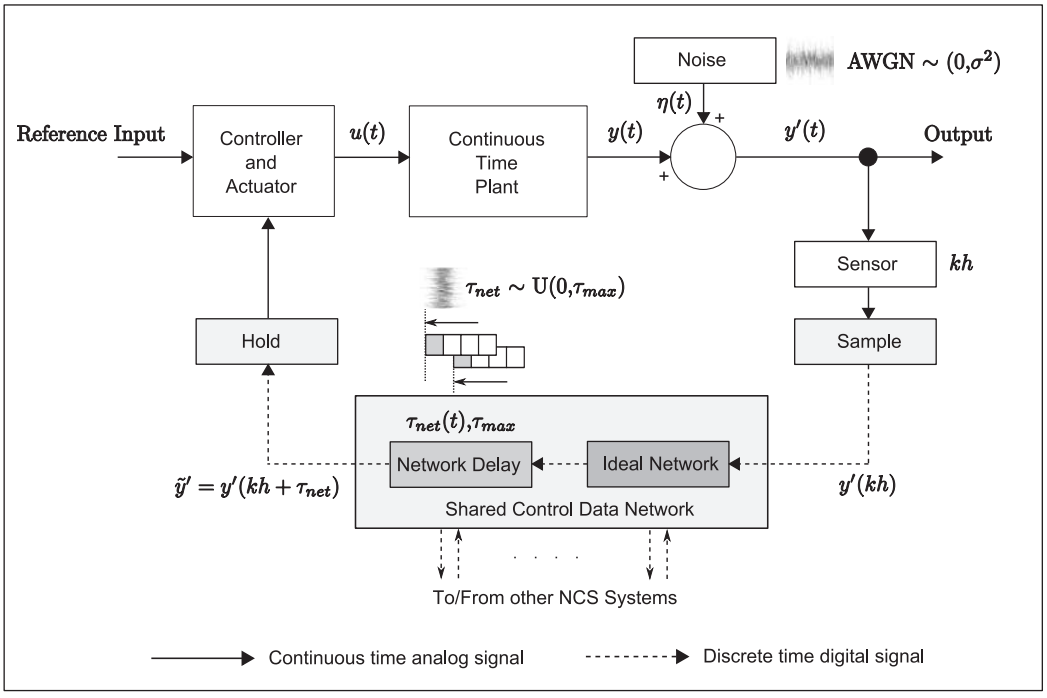

Figure 6. Topology of the NCS hybrid system example.

The system is affected by two stochastic processes. First, the continuous-time output signal of the plant to be controlled is perturbed by a continuous random process modeled as additive white Gaussian noise (AWGN) that represents the perturbation to be compensated by the controller. Also, a uniformly distributed random process models the varying network-induced delays affecting the network data packets carrying information from the sensor to the controller.

The continuous-time part of the model consists on the input reference, the controller, the actuator, the plant, the AWGN perturbation added at the plant's output signal (before it is sensed) and the sensor. The controller/actuator subsystem is a continuous-time event-driven component that calculates control signals and acts on the plant inputs whenever it receives new sensing information from the shared control network.

The discrete-time part of the model is composed of the sensor, the control network, and the respective analog-to-digital (sample) and digital-to-analog (hold) signal converters. Through this loop the noisy output of the plant is sensed and fed back on a periodic clock-driven basis affected by a random, upper-bounded, extra time delay. This random delay represents the resulting superposition of all the possible sources of delay found in the data network (e.g. queuing effects, access to physical medium, packet processing time, priority policies, etc.) owing to the fact that the control network is a shared, bandwidth-limited resource, with other NCSs.

The feedback closed-loop is composed of the sensor, the control network and the controller/actuator path, and is a hybrid loop that combines continuous-time and discrete-time signals with clock-driven and event-driven components.

We defined this system using STDEVS and implemented it in the discrete-event-based PowerDEVS simulator. We also implemented the same system in the well-known Matlab–Simulink discrete-time-based simulator. A control-related cost function is defined and then studied under several control network conditions by running simulations with both tools and then comparing the results.

7.3.1 NCS System Specification

The reference input value to be tracked by the system output is $Ref = 1$. The continuous-time plant is described by means of its transfer function, which is a mathematical representation (in terms of frequency) of the relation between the input and the output of an LTI system. The transfer function $G_P(s) = Y(s)/U(s)$ in the Laplace domain is a linear mapping of the Laplace transform of the input $U(s) = \mathcal{L}(u(t))$, to the Laplace transform of the output $Y(s) = \mathcal{L}(y(t))$, with s being the complex frequency of the system.

In our NCS system, the transfer function of the plant in the Laplace domain is $G_P(s) = 1/(s^2 + 0.8s)$ which is an unstable system at open loop (i.e. no feedback loop

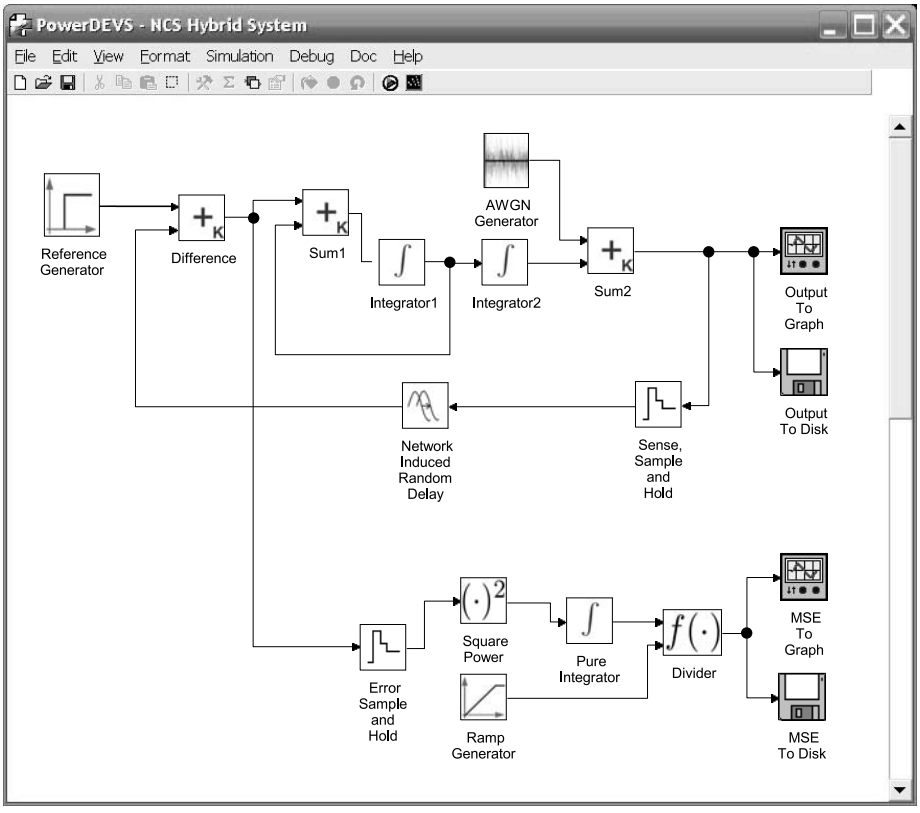

Figure 7. NCS PowerDEVS model.

between input and output) and stabilizes under unity feedback closed-loop conditions (which is the case of our system). At the output of the plant, the AWGN $\eta(t)$ has mean zero and variance $v_\eta = 0.001$. The sampling period for the discrete-time components is $h = 1$ seconds, and the network induced random delay is uniformly distributed according to $\tau_{\text{net}}(s) \sim U(0, \tau_{\max})$. Also, the delay is constrained with $\tau_{\max} < h$, so no queuing situation can happen at the component that models random network delay.

7.3.2 STDEVS NCS Model

In this section we show an implementation of the NCS system with PowerDEVS. In Figure 7 we show the block diagram of the STDEVS NCS model, including a branch (at the bottom) that calculates the cost function used for results analysis (see Section 7.3.6). We take as a reference the conceptual topology of the NCS system in Figure 6, and for each part of the topology we describe its functional match with an STDEVS component in the block diagram of Figure 7. The reference input is provided by the *Reference Generator* component with a constant output value of one. The controller and actuator subsystems are simply resolved by the *Difference* component, as it is a simple unity feedback control loop. The continuous time plant is composed of the *Integrator1*, *Integrator2* and *Sum1*; with the additive noise at its output signal provided by the *AWGN Generator* and *Sum2* components. The functionalities of sensor, sample and hold are resolved by a single *Sense, Sample and Hold* component. Finally, the shared control data network is modeled by a component that implements the network delay functionality, called the *Network Induced Random Delay* component.

Those components that raise special interest are the continuous-time integrators and the stochastic-related AWGN and random delay generators. In the case of the integrators, the QSS (quantized state systems) method [4] was used to approximate the continuous time subsystem[11]. In the following section we discuss the definition of those components that include stochastic behavior using STDEVS formal specification. The rest of the components defined as deterministic DEVS specifications are omitted for brevity, and can be found in [25].

11. QSS methods discretize continuous systems (ordinary differential equations) based on the quantization of the state variables. The resulting systems are equivalents to DEVS models.

7.3.3 Formal Specification for Stochastic Components

The components to be described are the *AWGN generator* (AWGN) and the *Network Induced Random Delay* generator (NIRD).

As we did in our first example (Section 7.2) for STDEVS specifications, we can rely on Theorem 1 and refer only to the measurable DEVS-RND form of the components that have their stochastic behavior expressed by RND() functions in their transition functions (here represented by the random variable r). Again, we can be confident of this because we know in advance that the stochastic models to be created are defined on finite-dimensional probability spaces, thus yielding measurable DEVS-RND models. This simplifies the modeling process avoiding the need to define the stochastic functions $\mathcal{G}_{\text{int}}$, $\mathcal{G}_{\text{ext}}$, P_{int}, P_{ext}.

Nevertheless, for the sake of completeness, we cover the STDEVS formulation for the AWGN component, and rely on Theorem 1 for the NIRD component.

AWGN Generator The AWGN component has no inputs and one output port through which it delivers normally distributed real-valued numbers representing the noise signal level $\eta(t)$.

The signals $\eta(t)$ have to be obtained from a continuous Gaussian distribution describing the probability $P(\eta \leq \varphi)$, where φ is a real value representing all of the possible noise levels, with $-\infty < \varphi < \infty$. This description is as follows:

$$P(\eta \leq \varphi) = \Phi_{\mu,\upsilon}(\varphi)$$

$$\Phi_{\mu,\upsilon}(\varphi) = \frac{1}{\sqrt{\upsilon}\sqrt{2\pi}} \int_{-\infty}^{\varphi} \exp\left[-\left(\frac{(u-\mu)^2}{2\upsilon}\right)\right] du, \quad \varphi \in \Re$$

where $\Phi_{\mu,\upsilon}$ is the cumulative distribution function of the signal, with μ and υ the mean expected value and the variance of the probability distribution, respectively. With this information we can start defining the stochastic components for the STDEVS model of the AWGN generator:

- $\mathcal{G}_{\text{int}}(s) = \{B_\varphi \mid -\infty < \varphi < \infty\}$, with $B_\varphi = (-\infty, \varphi$;
- $P_{\text{int}}(s, G) = P_{\text{int}}(s, B_\varphi) = \Phi_{\mu,\upsilon}(\varphi)$, $G \in \mathcal{G}_{\text{int}}$.

As we can see, the stochastic description of the noise level is mapped directly to the function P_{int} through the corresponding cumulative distribution function.

Now the STDEVS definition for AWGN,

$$M_{\text{ST}}^{\text{AWGN}} = (X, Y, S, \mathcal{G}_{\text{int}}, \mathcal{G}_{\text{ext}}, P_{\text{int}}, P_{\text{ext}}, \lambda, ta)$$

can be completed defining the state $s = (\eta, \sigma)$ (where η represents the noise level and σ is the time advance), and the deterministic components:

- $X = \oslash$, $Y = \Re \times \{out_1\}$;
- $S = \Re \times \Re_0^+$;
- $\lambda(\eta, \sigma) = (\eta, out_1)$;
- $ta(\eta, \sigma) = \sigma$.

As only internal transitions are possible, we do not need to define $\mathcal{G}_{\text{ext}}$, P_{ext}.

Now, we advance to the *Activity* (2) of our process, and pick a mathematical procedure that allows us to obtain successive Gaussian-distributed η values by manipulating uniform RND() outcomes. We want to find the measurable DEVS-RND formulation of our STDEVS AWGN model.

We denote the noise mean expected value as μ_η and the noise variance as υ_η for the noise level $\eta(t)$, and model them as external parameters.

The model can be defined in its measurable DEVS-RND form as follows:

$$M_D^{\text{AWGN}} = (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta)$$

where

- $X = \oslash$;
- $Y = \Re \times \{out_1\}$;
- $S = \Re \times \Re_0^+$;
- $\lambda(\eta, \sigma) = (\eta, out_1)$;
- $ta(\eta, \sigma) = \sigma$.

The internal transition is

$$\delta_{\text{int}}((\eta, \sigma), r) = (\tilde{\eta}, h)$$

with

$$\begin{cases} \tilde{\eta} = \left[\sqrt{-2\log(r_1)}\cos(2\pi r_2)\right]\sqrt{\upsilon_\eta} + \mu_\eta \\ r \in \Re^2, \quad r \sim U(0,1)^2, \quad r = (r1, r2) \\ h = \text{sampling interval} \\ \mu_\eta = \text{noise mean expected value} \\ \upsilon_\eta = \text{noise variance} \end{cases}$$

The external transition function will be $\delta_{\text{ext}}(s, e, x, r) = \oslash$ independent of r.

In this case, the calculation of the successive values of $\eta(t)$ is solved in δ_{int} using the Box–Müller transform [26],

a well-established and satisfactorily accurate[12] procedure for generating independent standard normally distributed random numbers given a source of uniformly distributed random numbers.

Network Induced Random Delay We give the measurable DEVS-RND definition for M_D^{NIRD}:

$$M_D^{\text{NIRD}} = (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta)$$

The state has the form $s = (y', \sigma)$, where $y' \in \Re$ represents the sampled version of the output signal with noise. Then,

$$\begin{cases} X = \Re \times \{inp_1\} \\ Y = \Re \times \{out_1\} \\ S = \Re \times \Re_0^+ \\ \lambda(y', \sigma) = (y', out_1) \\ ta(y, \sigma) = \sigma \end{cases}$$

The internal transition will be

$$\delta_{\text{int}}((y', \sigma), r) = (y', \infty)$$

independent of r, and the external transition function is

$$\delta_{\text{ext}}((y', \sigma), e, (x, inp_1), r) = (x, \tilde{\sigma})$$

with

$$\begin{cases} \tilde{\sigma} = r \cdot \tau_{\max} \\ r \in \Re, \quad r \sim U(0, 1) \\ \tau_{\max} = \text{network delay upper bound} \end{cases}$$

This definition inherently implies preemption. If an external transition is triggered by the arrival of the $(k+1)$th sample before the lifetime $ta(y_k, \sigma_k)$ of the kth sample has elapsed, the state will switch immediately from s_k to $s_{k+1} = \delta_{\text{ext}}(\cdot) = (x_{k+1}, \sigma_{\tilde{k+1}})$, thus preempting the kth sample processing.

Nevertheless, as stated before, because of the condition $\tau_{\max} < h$ imposed for $\tau_{\text{net}}(t)$ for all t there is no chance that the $(k+1)$th sample can arrive at this component before the delay action is fully completed for the kth sample.

7.3.4 Simulations and Results

In this section we show a comparison between the simulation results obtained with Matlab and PowerDEVS. For both models, we swept the simulation parameter $\tau_{\max}$ from 0.1 to 0.8 seconds with increments of 0.1 seconds. Then, given that the sampling period is $h = 1$ second, the worst additive network induced delay considered is 80% of the system digital clock period (for $0.8h < \tau_{\max} < h$ we verified an excessive variance of the cost function values as the system tends to be unstable, therefore these points were excluded as long as they do not offer any interesting information for our model comparison purposes).

7.3.5 Matlab Model

In the Matlab implementation of the model we used the standard building blocks provided by the Simulink *Continuous*, *Discrete*, and *Sources* libraries. We make some remarks about the components of most interest, and further details can be found in [28].

The *Continuous-time Plant* $G_p = 1/(s^2 + 0.8s)$ is described by a *Transfer Function* block. As mentioned before, the *Controller and Actuator* of this system is simply described by a subtraction block that calculates the difference between the *Reference* signal and the delayed sampled version of the noisy plant output (i.e. the error signal).

For the *AWGN Generator* we used the *Band-limited White Noise* block, which produces a random sequence of numbers with a selectable correlation time t_c, that can simulate the effect of white noise if t_c is chosen to be much smaller than the shortest time constant (i.e. the inverse of the real part of the fastest eigenvalue) of the system. We followed the rule of thumb suggested for accurate simulations and set $t_c = 0.1$ so that $t_c < 2\pi/100 f_{\max}$ holds, with $f_{\max}$ expressed in radians per second. While the covariance of true white noise is infinite, the approximation used in this block (because of the conversion from a continuous power spectral density to a discrete noise covariance) has the property that the covariance of the block output (cov) is the *NoisePower* parameter of the block divided by t_c. Accordingly we set $NoisePower = cov * t_c$, which for the case of the AWGN is $NoisePower = v_\eta * t_c = 0.001 t_c = 0.0001$ given $v_\eta = cov_\eta$ holds for zero mean normal distributions.

For the *Network Random Delay* generator we used the *Uniform Random Number* block for producing a random strip of numbers uniformly distributed between zero and the upper bound network delay parameter $\tau_{\max}$.

The clock-rate of the discrete-time part of the model is imposed by a *Zero-order Hold* block that covers the functions of the *Sample* and the *Hold* components of the conceptual model in Figure 6.

7.3.6 Cost Function

Cost functions are usually defined to give a measure of the control *Quality of Performance* (QoP). Popular cost functions (also referred as *performance criteria*) such as

12. The Box–Müller transform is only one of several possible approximation approaches [27, 20], and is attached to particular advantages and disadvantages for its numerical implementation. This analysis is out of the scope of the present work.

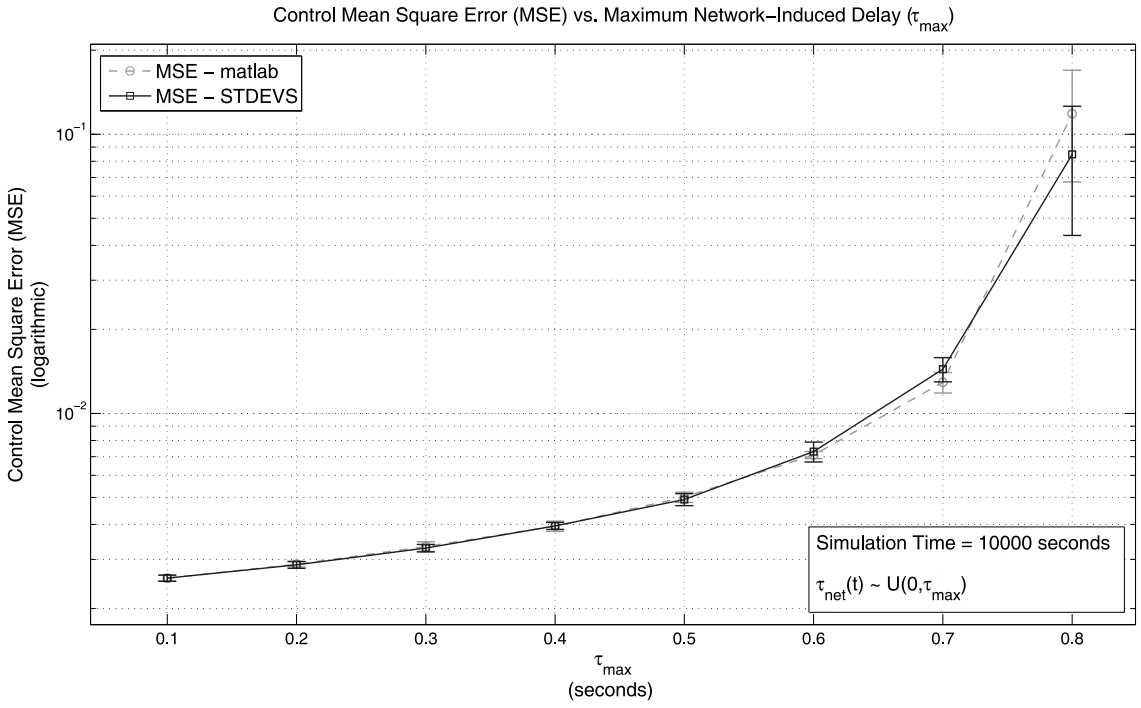

Figure 8. NCS hybrid model simulation results for Matlab and PowerDEVS.

the *Integral Time Absolute Error* (ITAE) and *Integral Absolute Error* (IAE) are calculated based on some form of treatment of the error signal along the simulation time [29].

We used the *Mean Squared Error* (MSE) cost function, which will help us to understand and compare the system simulation results for different network delay conditions by means of its system control performance. Its mathematical formula is as follows:

$$\text{MSE} = \frac{1}{T}\int_{t_0}^{t_f} e(t)^2\,dt$$

where t_0 and t_f are the initial and final times of the simulation period of length $T = t_f - t_0$ and $e(t)$ is the error between the reference value and the sampled and delayed version of the plant output trajectory (see Figure 7).

7.3.7 Results Analysis

Simulation results are shown in Figure 8, where simulations were run for $t_f = 10{,}000$ seconds and each point in the figure represents the MSE mean value and dispersion bars for 15 runs. We verify that the two curves are closely matched, thus validating the confidence and usefulness of STDEVS applied to a NCS hybrid model case.

However, there is a very important advantage of the STDEVS methodology over Matlab's discrete-time-based simulation in terms of robustness and computational effort of the results.

Regarding robustness, when using Matlab, we had to adjust the numerical integration method tolerance to a value of 1×10^{-6} (we used the ode45 method). Otherwise, the results had an unacceptable error. This is due to the hybrid nature of the problem, which combines discrete-event, discrete-time and continuous-time dynamics. When using STDEVS, we had to make no adjustments to the simulation parameters to produce reliable results. Thus, the efficient treatment of events becomes crucial.

Regarding computational effort, when using PowerDEVS (where all of the simulation is run under DEVS), we obtained faster simulations than Matlab by a speedup factor of about 5.7 times. This figure was obtained in a comparison study [28] where we ran and measured both simulators under controlled conditions. The simulation platform consisted of a Dell Inspiron 9400 computer with an Intel Core 2 Duo T7200 (2 GHz/667 MHz FSB/Family6/Model15/Stepping6) processor, 1 GB (DDR2/667 MHz) of memory and Windows XP Professional SP2 operating system. The performance studies were run for different combinations of possible configurations. We compared the simulator's processes both in normal priority and real-time priority and changed their processor core affinity to 1 and 2 cores alternatively. In all cases, for a virtual simulation time of $T = 50{,}000$ seconds we obtained similar speedup results, with wall–lock simulation times of about

$T_{\text{Matlab}} = 21.5$ seconds for Matlab and $T_{\text{PowerDEVS}} = 3.75$ seconds for PowerDEVS. The DEVS-based simulation speedup appears essentially because the numerical methods that approximate the continuous part (we used the QSS3 method) are particularly efficient to handle discontinuities and to integrate this type of hybrid system [30].

In consequence, the DEVS-based simulations are faster and more reliable than those based on time discretization.

8. Conclusions

We have presented a novel formalism for describing stochastic discrete event systems. Based on the system-theoretical approach of DEVS and making use of probability spaces theory, STDEVS provides a formal framework for modeling and simulation of generalized non-deterministic discrete event systems.

STDEVS is a general and comprehensive modeling formalism that provides the ability to represent stochastic behavior over measurable infinite-dimensional spaces, along with the ability to represent the traditional stochastic discrete event formalisms used widely in many applications (e.g. Markov chains, queuing networks, stochastic Petri nets, etc.). Owing to its DEVS-based roots, STDEVS is also a system-theoretic-based representation, which provides unique multimodeling advantages for the specification of hybrid systems, and important simulation performance enhancements when dealing with heavily discontinuous hybrid systems.

Next steps will be oriented to developing STDEVS-based libraries in PowerDEVS and CD++ for modeling and simulation of general purpose stochastic systems. The study of the implications of the STDEVS definition into the parallel DEVS formalism [31] will be also part of our future work.

9. Acknowledgement

This work was partially funded research grant José A. Estenssoro 2006 issued by Fundacion YPF Argentina.

10. References

[1] Zeigler, B. 1976. *Theory of Modeling and Simulation*. John Wiley & Sons, New York.

[2] Zeigler, B., T.G. Kim and H. Praehofer. 2000. *Theory of Modeling and Simulation*, 2nd edition. Academic Press, New York.

[3] Zeigler, B., and S. Vahie. 1993. Devs formalism and methodology: unity of conception/diversity of application. In *Proceedings of the 25th Winter Simulation Conference*, Los Angeles, CA, pp. 573–579.

[4] Cellier, F.E. and E. Kofman. 2006. *Continuous System Simulation*. Springer, New York.

[5] Giambiasi, N., B. Escude and S. Ghosh. 2000. GDEVS: a generalized discrete event specification for accurate modeling of dynamic systems. *Transactions of the SCS*, 17(3): 120–134.

[6] Nutaro, J. 2003. *Parallel Discrete Event Simulation with Application to Continuous Systems*. PhD Thesis, The University of Arizona.

[7] Zeigler, B. and H. Sarjoughian. 2000. *Introduction to DEVS Modeling and Simulation with JAVA: A Simplified Approach to HLA-Compliant Distributed Simulations*. Arizona Center for Integrative Modeling and Simulation.

[8] Wainer, G., G. Christen and A. Dobniewski. 2001. Defining DEVS Models with the CD++ Toolkit. In *Proceedings of ESS2001*, Marseille, France. SCS Publisher, pp. 633–637.

[9] Filippi, J.B., M. Delhom and F. Bernardi. 2002. The JDEVS Environmental Modeling and Simulation Environment. In *Proceedings of IEMSS 2002*, Vol. 3, pp. 283–288.

[10] Kofman, E., M. Lapadula and E. Pagliero. 2003. *PowerDEVS: A DEVS Based Environment for Hybrid System Modeling and Simulation*. Technical Report LSD0306, LSD, UNR. Available at http://www.fceia.unr.edu.ar/∼kofman.

[11] Cutler, M.M. 1980 Discrete event simulation of stochastic and deterministic sequential machine models. In *Proceedings of the 12th Winter Conference on Simulation*, Orlando, FL, pp. 83–97.

[12] Cassandras, C. 1993. *Discrete Event Systems: Modeling and Performance Analysis*. Irwin and Aksen, Boston, MA.

[13] Ajmone Marsan, M., G. Balbo, G. Conte, S. Donatelli and G. Franceschinis. 1995. *Modelling with Generalized Stochastic Petri Nets*. John Wiley & Sons, Chichester.

[14] Aggarwal, S. 1975. *Ergodic Machines—Probabilistic and Approximate Homomorphic Simplifications*. PhD Thesis, The University of Michigan, Ann Arbor, MI.

[15] Melamed, B. 1976. *Analysis and Simplifications of Discrete Event Systems and Jackson Queuing Networks.* PhD Thesis, The University of Michigan, Ann Arbor, MI.

[16] Joslyn, C. 1996. The process theoretical approach to qualitative DEVS. In *Proceedings of the 7th Conference on AI, Simulation, and Planning in High Autonomy Systems (AIS'96)*, San Diego, CA, pp. 235–242.

[17] Gray, R. and L. Davisson. 2004. *An Introduction to Statistical Signal Processing*. Cambridge University Press, Cambridge.

[18] Billingsley, P. 1995 *Probability and Measure*, 3rd edition. John Wiley & Sons, New York.

[19] Vitali, G. 1905. *Sul problema della mesura dei gruppi di punti di una retta*. Bologna.

[20] Knuth, D.E. 1997. *Art of Computer Programming, Volume 2: Seminumerical Algorithms*, 3rd edition. Addison-Wesley Professional.

[21] Institute for Statistics of the Vienna University of Economics and Business Administration. 2009. ARVAG Project—Automatic nonuniform Random VAriate Generation. http://statistik.wu-wien.ac.at/arvag/.

[22] Kleinrock, L. 1975. *Queuing Systems, Vol. 1: Theory*. John Wiley & Sons, New York.

[23] Nilsson, J., B. Bernhardsson and B. Wittenmark. 1998. Stochastic analysis and control of real-time systems with random time delays. *Automatica*, 34(1): 57–64.

[24] Walsh, G.C. and H. Ye. 2001. Scheduling of networked control systems. *IEEE Control Systems Magazine*, 21(1): 57–65.

[25] Castro, R. and E. Kofman. 2008. *Discrete Event Modeling and Simulation of Networked Control Systems*. Technical Report LSD0808, LSD, UNR. Available at: http://www.fceia.unr.edu.ar/∼kofman.

[26] Box, G.E.P. and M.E. Müller. 1958. A note on the generation of random normal deviates. *The Annals of Mathematical Statistics*, 29(2): 610–611.

[27] Press, W.H., S.A. Teukolsky, W.T. Vetterling and B.P. Flannery. 2007. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, Cambridge.

[28] Castro, R. and E. Kofman. 2008. Simulation of NCS systems: a performance study of discrete event and discrete time tech-

niques. Technical Report LSD0809, LSD, UNR. Available at: http://www.fceia.unr.edu.ar/∼kofman.
[29] Franklin, G.F., J.D. Powell and A. Emami Naeini. 1994. *Feedback Control of Dynamic Systems*. Addison-Wesley, Reading, MA.
[30] Kofman, E. 2004. Discrete event simulation of hybrid systems. *SIAM Journal on Scientific Computing*, 25(5): 1771–1797.
[31] Chow, A.C.-H. 1996. Parallel DEVS: a parallel, hierarchical, modular modeling formalism and its distributed simulator. *Transactions of the Society for Computer Simulations International*, 13(2): 55–67.