Hybrid Modeling and Simulation of Discrete Event Systems

Myung Soo Ahn and Tag Gon Kim

Department of Electrical Engineering
Korea Advanced Institute of Science and Technology
373-1 Kusong-Dong,Yusong-Gu,Taejon 305-701, Korea

KEYWORDS

DEVS Formalism, Hybrid Modeling/Simulation, Model Transformation, Performance Evaluation

ABSTRACT

This paper seeks to a methodology to utilize both advantages from analytic and simulation modeling/simulation. For hybrid modeling, we propose a model transformation scheme which transforms selected simulation models into analytic ones as far as accuracy is preserved. A hybrid simulation algorithm which can simulate both simulation and analytic models in a combined manner with greatly reduced simulation time, is developed. The proposed methodology exploits the Zeigler's DEVS formalism and associated system theory. To show the efficiency, we demonstrate performance modeling and simulation of communication networks.

INTRODUCTION

There are two approaches to performance modeling of discrete event systems: an analytic model and a simulation model. Although analytic models have limitations in accuracy due to the unrealistic assumptions made, fast simulation experimentations are possible by employing appropriate numerical analysis techniques. On the other hand, simulation models have expressive means powerful enough to achieve high accuracy. However, simulation time for such models is extremely slow compared with that for analytic models due to some virtual management scheme embedded in a simulation algorithm. Thus, it is highly desirable to combine the advantages from both models in an unified framework[1].

The purpose of this paper is to develop a framework within which both accuracy in models and speed in simulation experiments are obtained. A hybrid model consists of analytic sub-models and simulation sub-models. An accurate analytic sub-model can be obtained by transformation of a simulation sub-model. More specifically, a discrete event model, specified by DEVS (Discrete EVent Systems specification) formalism, which meets a certaion condition has shown to be equivalent to a Markovian process in steady state. Simulation run only evaluates simulation sub-models and analytic sub-models are evaluated separately. To be correct, the evaluation result of analytic sub-models are added to that of simulation sub-models. Since a simulation cycle doesn't include analytic sub-models hybrid simulation can reduce simulation time markedly without losing accuracy.

We employ the DEVS formalism[6], which supports hierarchical modular descriptions of discrete event systems. Since our approach uses a single modeling formalism, it enables the modelers to develop models within the expressive formalism. Also, it is based on the sound mathematical foundations in the behavioral equivalence relation between the DEVS models and the transformed analytic models. To exemplify the proposed hybrid modeling and simulation framework, we demonstrate performance modeling and simulation of communication networks.

HYBRID MODELING OF DEVS MODELS

Zeigler's DEVS(Discrete Event System Specification) formalism specifies discrete event models in hierarchical manner based on the "closure under coupling" concept[6].

The DEVS Formalism

A set-theoretic formalism, the DEVS formalism specifies discrete event models in a hierarchical, modular form. Within the formalism, one must specify 1) the basic models from which larger ones are built, and 2) how these models are connected together in hierarchical fashion. A basic model, called an atomic model

(or atomic DEVS), has specification for dynamics of the model. An atomic model AM is specified by a 7-tuple:

$$AM =< X, S, Y, \delta_{int}, \delta_{ext}, \lambda, ta >$$

X : input events set;
S : sequential states set;
Y : output events set;
$\delta_{int} : S \rightarrow S$: internal transition function;
$\delta_{ext} : Q \times X \rightarrow S$: external transition function;
$\lambda : S \rightarrow Y$: output function;
$ta : S \rightarrow Real$: time advance function,

where $Q = \{(s, e) \mid s \in S, 0 \leq e \leq ta(s)\}$: total state of AM.

The second form of the model, called a coupled model (or coupled DEVS), defines how to couple (connect) several component models together to form a new model. This latter model can itself be employed as a component in a larger coupled model, thus giving rise to construction of complex models in hierarchical fashion. A coupled model CM is defined as[6]:

$$CM =< X, Y, M, EIC, EOC, IC, SELECT >$$

X : input events set;
Y : output events set;
M : DEVS components set;
$EIC \subseteq CM.IN \times M.IN$: external coupling;
$EOC \subseteq M.OUT \times CM.OUT$: external coupling;
$IC \subseteq M.OUT \times M.IN$: internal coupling;
$SELECT : 2^M - \emptyset \rightarrow M$: : tie-breaking selector,

where the extensions .IN and .OUT represent the input ports set and output ports set of respective DEVS models.

The three coupling relations, namely, EIC, EOC, and IC, specifies input/output connections between component models and the coupled model.

The abstract simulator introduced in [6] is a conceptual device capable of interpreting the dynamics of DEVS models. Two classes of the abstract simulator have been defined: simulator for the atomic DEVS and coordinator for the coupled DEVS.

Construction of Hybrid Models

We transform the DEVS model into a behaviorally equivalent analytic model to analyze the steady state behavior of a DEVS model without time-consuming simulation experiments. Classification of discrete event specification are useful in identifying behavioral characteristics of DEVS models. The concepts of active and passive DEVS are essential in model transformation[6].

Active DEVS Models : Generators

An active DEVS is one which it is not passive. Such a DEVS has at least one active state in which it can initiate activity of its own choosing by means of its self scheduling mechanism. A *generator* is a typical example of active DEVS which is input-free. Generators may be used to implement arrival processes among other classes of input segments to models. Since a generator controls the simulation length and workload, it can not be transformed into an analytic model.

Passive DEVS Models : Transducers

A DEVS is passive if its time advance function is identically infinite. Such a DEVS is not capable of initiating activity on its own since the time to its next internal transition is always infinity. A transducer is a special DEVS to gather statistics about model trajectories. If an atomic DEVS model has only one state and no output events set, the model is a transducer. Such a transducer model also can not be transformed into analytic one.

Passive DEVS Models : Finite Memory

More general classes of passive models are finite state DEVS and infinite state DEVS models. A finite state DEVS is a model that its state space is finite. Thus, it saves a finite number of past inputs and outputs. Such a finite state DEVS can be transformed into an equivalent Continuous Time Markov Chain.

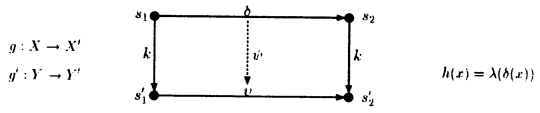
While the state transition is transformed, the input and output events sets are preserved in the transformed analytic model. The transformed analytic models can be represented using a mathematical structure such as the one used to describe atomic DEVS models. The structure for analytic model is defined by a 5-tuple :

$$TM =< X, Y, S, \Upsilon, H >$$

X : input events set;
Y : output events set;
S : states set;

Q : state transition rates matrix;
$H : X \rightarrow Y$: I/O translation function;

As shown below, the transformation algorithm is a process for finding two mappings $\psi, h(x)$ which are associated with the transition rates and the output function, respectively.



Passive DEVS Models : Infinite Memory

A infinite memory DEVS model is a queueing model which can save a number of incoming jobs. Infinite state DEVS models are more classified into FIFO queues and Priority Queue based on the order in which the incoming events are processed. It is impossible to automatically transform a priority queue into an analytic queueing model because the model's behavior depends on the contents of events for determining the priority.

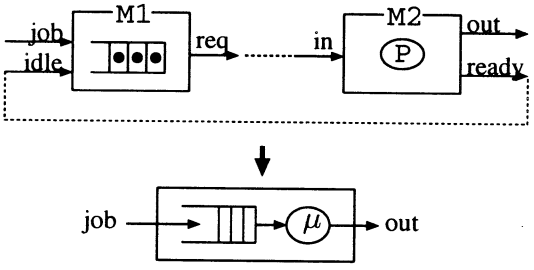


Figure 1: Aggregation of Two Models into a Single Model

Almost couplings between component models are queueing couplings. Figure 1 shows this class of couplings and state transitions. Using the identification, steady state rates for each event can be determined. The transformed queueing models are specified as :

$QM =< X, Y, \mu, \lambda, H >$

X : input events set;
Y : output events set;
μ : service rate;
λ : arrival rates of input events;
$H : X \rightarrow Y$: I/O translation function;

Group	Instances	Transformability
$X = 0$	Generators	Do Not Transform
$Y = 0$	Transducers	Do Not Transform
$X \neq 0 \wedge Y \neq 0$	Finite Memory DEVSs Infinite DEVS (FIFO) Infinite DEVS (Priority)	Transform into CTMCs Transform into Queues Do Not Transform

Figure 2: Classification of Models and Their Transformability

Figure 2 summarizes the classification of atomic DEVS and transformability of each class.

HYBRID SIMULATION METHODOLOGY

To simulate both simulation models and analytic models in a combined manner, we develop a hybrid simulation algorithm satisfying the requirements of interface consistency between models and computation efficiency. Interface consistency dictates that simulation of analytic models should not disturb the event-driven nature of simulation models. We extend the abstract simulator algorithms for DEVS models by adding an analyzer for transformed analytic models.

We apply the following three propositions for developing the simulation algorithm. Let λ be a Poisson process of rate λ and M be a Markovian transformation of incoming process.

Proposition 1 : *Rule of Preservation* $[M(\lambda) = \lambda]$
The departure process from a Markovian queue is the same as the arriving process.

Proposition 2 : *Rule of Superposition* $[\lambda_1 + \lambda_2 = \lambda]$
A superposition of two Poisson processes results in a Poisson process.

Proposition 3 : *Rule of Split* $[\lambda = p \cdot \lambda + (1 - p) \cdot \lambda]$
p is a constant which is less than or equal to 1. Splitting a Poisson process into two outgoing processes based on a constant probability results in two Poisson process.

From the above propositions, we can derive the following facts. First, the behavior of an analytic model in the steady state depends only on the rates of incoming events. If a model has the Markov property,

Initialization : $t_N = \infty$;
Simulation:
When receive an input (x, t)
done := false;
update Q or $\lambda(x)$;
$Y := H(x)$;
$t_N := \infty$;
done := true;
end when
Termination : report the performance of model;

Figure 3: Behavior Outline of Analyzer

the rates of incoming events and output events are the same. Since we transform DEVS models into analytic models only when the models satisfy the Markov property, it is possible to immediately route input events of an analytic model to other models. This routing policy does not change the rates of events between models.

Concludingly, analytic models need not to hold the incoming events for the processing times and can directly send them out to other models. If so, the performance of DEVS models will be correct. Of course, the performance of a system should be adjusted after the simulation by adding the performance of analytic models to the simulation result.

For hybrid simulation, we extend the abstract simulator algorithms of the DEVS formalism to include the analyzers for analytic models. Figure 3 shows job of analyzer associated with a transformed analytic model.

It is important to note that the analyzer always returns ∞ as the next schedule time. This means that the analytic models do not act as generators. They only act as processors. As shown in Figure 3, the behavior of analyzer and analytic models is very simple compared with that of simulator and atomic DEVS models.

HDEVSim++ is an object-oriented environment which supports modelers to develop discrete event models using the DEVS formalism. It extends the DEVSim++ environment[2]. HDEVSim++ can automatically transform simulation models into equivalent analytic ones. The hybrid simulation engine of HDEVSim++ realizes the above simulation algorithm. The environment is a result of the combination of two powerful frameworks for systems simulation: the model transformation method and the extended abstract simulator algorithms.

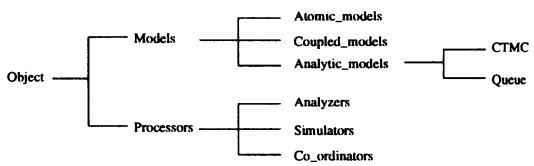


Figure 4: Class Hierarchy of HDEVSim++

As shown in Figure 4, HDEVSim++ defines classes for modeling and those for simulation separately. Thus, it can be easily evolved by developing new classes. Hybrid simulation of models developed in HDEVSim++ is managed by means of message passings among three subclasses of *Processors* : *Simulators*, *Co_ordinators*, and *Analyzers*.

After developing and validating all models including those for the experimental frame, simulation experiments can be started. HDEVSim++ terminates the simulation when all models are passivate, that is, the next scheduling times of all models are infinity. Upon terminating a simulation, the coordinator corresponding to the outmost coupled model calls the children for evaluating performance of the associated models.

EXAMPLE AND RESULTS

For evaluating the computation efficiency of the proposed approach, we exemplify the simulation experiments of communication networks.

10-Node Networks with Priority Queues

We consider the 10-node network with priority servers of Figure 5. A similar network topology can be found in [4]. The packets in the network fall into one of two classes, which have different arrival distributions and service times. In [4], finite buffer sizes at all nodes are assumed. But, we consider the infinite buffer sizes. Such a network can be used for modeling of large number of real-world systems such as manufacturing and communication, and traffic networks.

To evaluate the efficiency of two simulation methods, extensive simulation experiments for the developed model in HDEVSim++ were performed. Figure 6 shows the simulation times of two simulation methods for the 10-node network with varying workloads. By transforming 40% of models (queues in the dotted

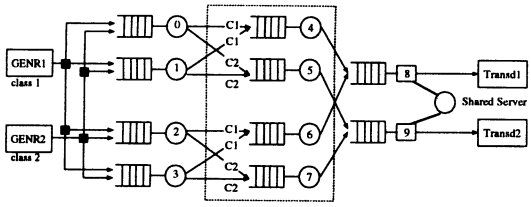


Figure 5: A 10-node Network with Priority Servers

box), we can improve the simulation times about 30%. The effect of improvement is more significant when the systems become more complex or when simulation experiments run long enough to obtain the statistical significance of the results.

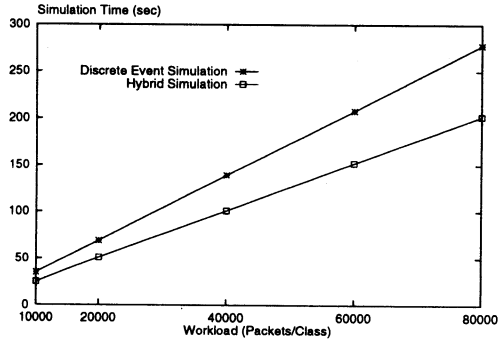


Figure 6: Simulation Times for Varying Workload

Although the proposed approach improves the computation overhead of discrete event simulation, we need to compare the results of hybrid method with those of simulation for validating the accuracy. For comparison, we measured the average processing times for packets with varying the workload. As shown in Table 7, the proposed approach accurately estimates the system performance. The % errors are less than 1 % for all cases.

Hypercube Networks

To evaluate the performance of message routing in a hypercube network, we modeled the hypercube model in HDEVSim++ and simulated it using the discrete event simulation. After verifying the developed model, we applied the model transformation method and then performed fast simulation experiment within the hybrid simulation framework of

Workload (Packets/Class)	Class 1			Class2		
	DEVS	Hybrid	% Error	DEVS	Hybrid	% Error
10000	3076	3070	-0.20	3157	3188	0.97
20000	3078	3083	0.16	3150	3166	0.51
40000	3087	3092	0.16	3160	3169	0.28
60000	3092	3089	-0.10	3164	3181	0.53
80000	3084	3081	-0.10	3161	3182	0.66

Figure 7: Average Processing Times

HDEVSim++. The result of transformation is shown in Figure 8. All atomic models in the communication module of each node are transformed into analytic ones.

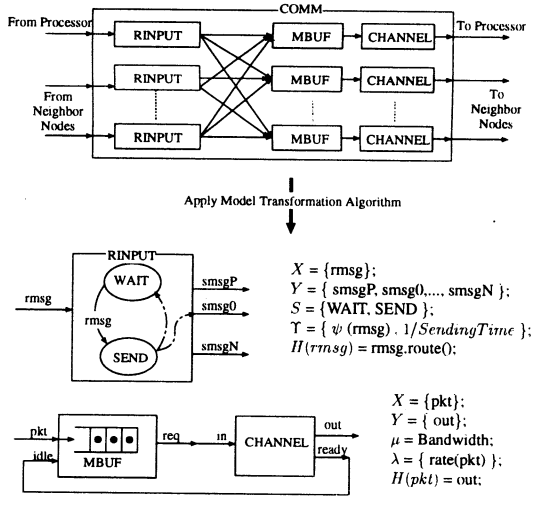


Figure 8: Transformation of Communication Module

Various test configurations were examined. Figure 9 shows the simulation times of two types of models for different node sizes. The packet generation rate at a node is set to 1000 messages/sec. As shown in the figure, computation costs for discrete event simulation rapidly grow when the dimension of cube increases. But, the hybrid simulation does not show the rapid growing of computation costs.

Figure 10 compares the accuracy of two methods. The differences between two models are very small. Thus, our approach is well suited for performance

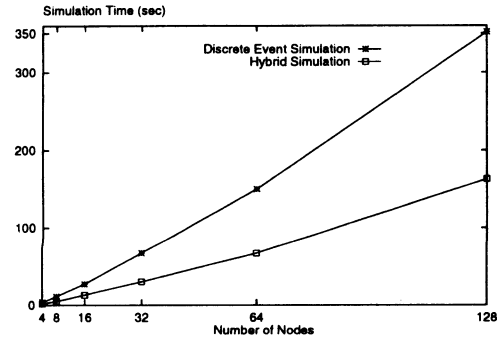


Figure 9: Simulation Times for Varying Number of Nodes

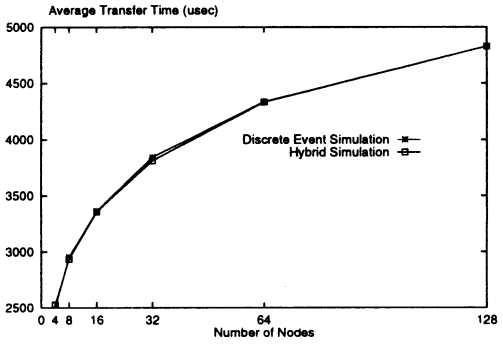


Figure 10: Average Processing Times

evaluation of systems whose behaviors satisfy the Markovian property.

SUMMARY

We have proposed a hybrid modeling and simulation framework for high-speed simulation without losing the accuracy of discrete event simulation. The recognition that atomic DEVS models can be transformed into equivalent analytic ones has led to the development of a new hybrid modeling methodology within a unified modeling framework. For hybrid simulation of the transformed analytic models and DEVS models, we developed a hybrid simulation environment which is capable of simulating both analytic and simulation models in a combined manner. To verify the proposed approach, we modeled communication networks and compared the results with those obtained from simulation experiments with only discrete event simulation models. The results show that our approach can accurately simulate the behavior of a system with greatly reduced simulation time.

References

[1] M. S. Ahn and T. G. Kim, "Analysis on steady state behavior of devs models," in *Proc. of 4th Annual Conf. on AI, Simulation, and Planning in High Autonomy Systems(AIS '93)*, pp. 142–147, Sept. 1993.

[2] M. S. Ahn and T. G. Kim, "Devsim++ user's manual," Tech. Rep. TR-CORE-94-1, Core Lab., EE, KAIST, 1994.

[3] C. Cassandras, *Discrete Event Systems : Modeling and Performance Analysis*. Richard D. Irwin, Inc., and Aksen Associates, Inc., 1993.

[4] C. Chen and Y. Ho, "An approximation approach of the standard clock method for general discrete-event simulation," *IEEE Transactions on Control Systems Technology*, vol. 3, pp. 309–317, Sept. 1995.

[5] E. Yucesan and L. Schruben, "Structural and behavioral equivalence of simulation models," *ACM Transactions on Modeling and Computer Simulation*, vol. 2, pp. 82–103, Jan. 1992.

[6] B. Zeigler, *Multifacetted Modeling and Discrete Event Simulation*. Orlando, FL.: Academic Press, 1984.