Proceedings of the 1994 Winter Simulation Conference
ed. J. D. Tew, S. Manivannan, D. A. Sadowski, and A. F. Seila

PARALLEL DEVS: A PARALLEL, HIERARCHICAL, MODULAR MODELING FORMALISM

Alex ChungHen Chow
Object Technology Products
IBM Corp.
Austin, TX 78758

Bernard P. Zeigler
Department of Electrical and Computer Engineering
The University of Arizona
Tucson, AZ 85721

ABSTRACT

We present a revision of the hierarchical, modular Discrete Event System Specification (*DEVS*) modeling formalism. The revision distinguishes between transition collisions and ordinary external events in the external transition function of *DEVS* models. Such separation enables us to extend the modeling capability of the collisions. The revision also does away with the necessity for tie-breaking of simultaneously scheduled events, as embodied in the *select* function. The latter is replaced by a well-defined and consistent formal construct that allows all transitions to be simultaneously activated. The revision provides a modeler with both conceptual and parallel-execution benefits.

1 INTRODUCTION

The *Discrete Event System Specification(DEVS)* formalism was introduced in the early 70s and later extended to enable constructing discrete event simulation models in a hierarchical, modular manner (Zeigler 1976, 1984). Not only does it provide a powerful modeling methodology but also a framework for model behavior generation via its abstract simulator concepts (Zeigler, 1990).

Hierarchical modeling capability is increasingly being recognized as the predominant modeling paradigm for future simulation developments. Sargent (1993) lists the advantages of such capability as: reduction in model development time, support for reuse of a database of models, and aid in model verification and validation. Although none of the major existing simulation languages support hierarchical modeling, research environments have emerged over the last few years (Zeigler 1990, Praehofer 1993, Rozenblit and Janknski 1990, Kim 1994) and commercial simulation vendors are showing interest (Pegden and Davis 1992). Distributed and parallel simulation has also received increasing interest as simulations become more time consuming and geographically distributed (Fujimoto 1990, Lubachevesky et al. 1989, Preiss 1991).

As with all modeling methodologies, the *DEVS* simulations are prone to manifest behaviors that are difficult to conceptualize as real life phenomena. Radiya and Sargent (1994) introduced a logic-based foundation to study the semantics of these mappings. Multiple state transitions at the same simulation time lead to such non-intuitive behavior. We use the term *transition collisions* to represent mutually interfering simultaneous events. Such collisions can happen either between internal and external transitions (simply referred to as *collisions*) or between multiple external transitions (referred to as simultaneous events). The latter can occur when a *DEVS* model is constructed by coupling component models together. In either case, the model behavior generated is tricky to follow and may easily deviate from that intended if the collisions are not handled properly.

In discrete event languages, it is a modeler's responsibility to understand the model coupling and collisions and to modify a model to make sure that the collisions are well handled and explainable. In this article, we emphasize the handling of the collisions and propose a generalized *Parallel DEVS* specification intended to facilitate proper modeling of both kinds of the collisions within coupled models. As an important result, the generalization exhibits increased parallelism that can be exploited in a parallel/distributed simulation.

The article reviews the *DEVS* formalism and discusses the motivation behind the revision. The *Parallel DEVS* formalism is then introduced and shown to provide the means for properly handling the collision problems. The formalism is also shown to be closed under coupling, thus preserving hierarchical, modular construction properties. The construct leads to the definition of an *Parallel DEVS* abstract simulator which correctly implements the formalism and exploits the increased parallelism.

2 THE ORIGINAL DEVS FORMALISM

DEVS is the formalism that allows a modeler to specify a hierarchically decomposable system as a discrete event model that can be later simulated by a simulation engine. Several *DEVS* formalisms evolved from the original one to introduce specializations for different purposes. This section reviews the original formalism and discusses the reasons behind the revisions.

The original *DEVS* model is a structure:

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

X: a set of external events.
S: a set of sequential states.
Y: a set of output events.
$\delta_{int} : S \rightarrow S$: internal transition function.
$\delta_{ext} : Q \times X \rightarrow S$: external transition function,
where $Q = \{(s, e) | s \in S, 0 \leq e \leq ta(s)\}$ and
e is the elapsed time since last state transition.
$\lambda : S \rightarrow Y$: output function.
$ta : S \rightarrow R_{0 \rightarrow \infty}$: time advance function.

The *DEVS* model transits among the states, S, via its transition functions. When no external event occurs, the time of an internal state transition is determined by the ta applied to the current state. The new state of the model is determined by δ_{int} applied to the old state. Right before any internal transition, the model can generate an output event that depends on the state before the internal transition. A state transition can happen when an external input event occurs as well. The δ_{ext} determines the new state based on the current state, the time elapsed in this state, and the external event.

The transition sequences are illustrated in *Figure 1*. $s2 = \delta_{int}(s1)$, $s3 = \delta_{int}(s2)$, and etc. are the internal transitions when no external event occurs. $ta(s2)$, $ta(s3)$, and etc. determine the time of an internal state transition. $y1 = \lambda(s1)$, $y2 = \lambda(s2)$, and etc. are the output events generated before the internal transitions. When the external input event, $x1$, is received, the resulting external transition depends on the current state, $s4$, the elapsed time, $e4$, and the input event, $x1$, to determine a new state, $s5 = \delta_{ext}(s4, e4, x1)$.

A *coupled model* is a structure:

$$DN = < X_{self}, Y_{self}, D, \{M_i\}, \{I_i\}, \{Z_{i,j}\}, select >$$

D: a set of components.
for each i in D,

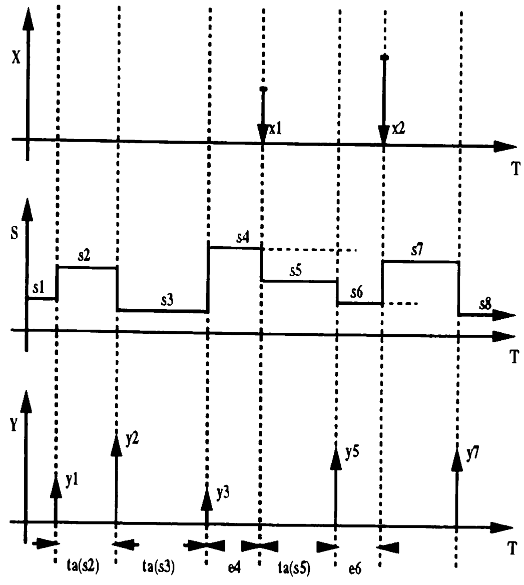

Figure 1: State Transitions of a DEVS Model

M_i is a component.
for each i in $D \cup \{self\}$,
I_i is the influencees of i.
for each j in I_i,
$Z_{i,j}$ is a function, the i-to-j output translation.
select is a function, the tie-breaker.

The structure is subject to the constraints that for each i in D,
$M_i = < X_i, S_i, Y_i, \delta_i, \lambda_i, ta_i >$,
I_i is a subset of $D \cup \{self\}$, i is not in I_i,
$Z_{self,j} : X_{self} \rightarrow X_j$,
$Z_{i,self} : Y_i \rightarrow Y_{self}$,
$Z_{i,j} : Y_i \rightarrow X_j$.
select : subset of $D \rightarrow D$,
such that for any non-empty subset E,
$select(E) \in E$.

A coupled model groups several *DEVS* models into a composite which can be regarded as yet another *DEVS* model. Due to this closure under coupling (Zeigler 1984), models can be constructed in hierarchical fashion from atomic *DEVS* models (A *DEVS* model that is not constructed by using a coupled model is called an *atomic model*.).

In the abstract simulator concept (Zeigler 1984), the simulation of the atomic and coupled models is carried out by processors called *simulators* and *coordinators* respectively. Basically, the simulation is triggered by receiving simulation messages, $(*, t)$ and (x, t), which in turn sequentially invoke outputs, external transitions and finally *schedule* new internal transitions at future times given by the ta functions.

In this *DEVS* structure and its serial *co-ordinator*, a modeler must employ the *select* function as a tie-breaker of simultaneously scheduled internal events. The formalism allows a model state to exist either at both the boundaries of the elapsed time interval: $e = 0$ and $e = ta(s)$, both of which represent the time of internal transition. In coupled models, ambiguity arises when an external event is received by a model at the same time of its scheduled internal transition — which elapsed time should be used by the external transition function, $e = 0$ or $e = ta(s)$? A *select* function solves this ambiguity by choosing, as active, one from the set of imminent components (those scheduled to make internal transitions at the current simulation time): for this selected component, $e = 0$; for the others, $e = ta(s)$.

Although reflecting the approaches of conventional serial simulation languages, this tie-breaking approach is a potential source of error. The serialization it engenders may not properly reflect the co-occurrence of events in the system being modeled, particularly among mutual influencing imminent components. Moreover, the serialization reduces the possible exploitation of parallelism among concurrent events. When interpreted in a parallel simulation environment, it is appropriate to parallelize the handling of simultaneously scheduled events to achieve the greater speedup.

2.1 The Extended *DEVS*

The *Extended DEVS(E-DEVS)* formalism (Wang 1992, 1993) attempts to exploit parallelism by removing the *select* function. However, in its place a new *order* function is added to the coupled model which serializes the execution of simultaneously received external events. Because of this arrangement, event queues are used to hold the external events arriving at the same time. The *select* function can be removed because the *E-DEVS* formalism predefines the behavior of a component model at a collision time to be $s = \delta_{ext}(\delta_{int}(s), 0, x)$. This means that if a collision occurs, the internal transition is always carried out first and the colliding external transition function is then applied to the new state $s_{new} = \delta_{int}(s)$, with $e = 0$.

However, the use of an *order* function is still unsatisfactory since a modeler still has to understand the interrelationship between the different external transitions to correctly choose an *order* function. Moreover, if an interdependency exists among two or more external transitions, the *order* function cannot correctly model the joint effect of these events. This limitation is similar to that of the *select* function it replaces.

In the *DEVS* formalism the input event set can be defined to be the power set (set of all subsets) of a set of events. The external transition function then regards the occurrence of a subset (simultaneous events) as a single event. Although both the original and the *E-DEVS* formalisms allow the event set to be a power set, the abstract simulators do not support this concept. They both treat the simultaneous events sent to a component in sequence. The parallelism among the simultaneous external events is not exploited.

From the above discussion, several properties are desirable for the newly revised *DEVS* formalism.

Collision Handling: The behavior of a collision must be controllable. Predefining a collision behavior is a limitation on the modeling capability that is not a necessary price to pay for parallelism.

Parallelism: The formalism must not use serialization function that prohibits possible concurrencies. The parallelism among the internal transitions and simultaneous events must be fully exploited.

Closure: The formalism must be closed under coupling and thereby, support hierarchical construction.

Hierarchical Consistency: The hierarchical construction must render a uniform behavior in that different hierarchical constructs of the same model must display the same behavior.

The *Parallel DEVS* formalism was developed in order to address these requirements.

3 THE PARALLEL DEVS

Based on the above discussion, we find that the key to meet these requirements is to properly handle collisions, i.e. the behavior when a component receives external events at the same time as its prescheduled internal transition. Previous solutions attempt to define the collision behavior implicitly either through the *select* function or by imposing the priority of an internal transition over a colliding external transition.

In contrast, the *Parallel DEVS(P-DEVS)* formalism structure proposed here enables a modeler to explicitly define the collision behavior by using the so-called confluent transition function, δ_{con}.

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

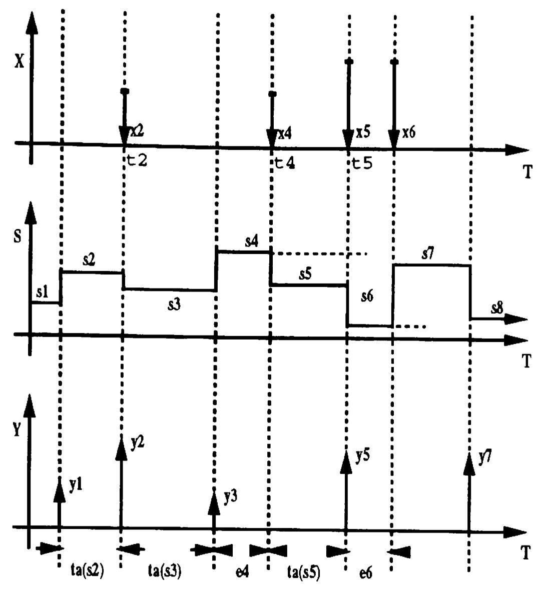

Figure 2: State Transitions of a Parallel DEVS Model

X: a set of input events.
S: a set of sequential states.
Y: a set of output events.
$\delta_{int} : S \rightarrow S$: internal transition function.
$\delta_{ext} : Q \times X^b \rightarrow S$: external transition function,
X^b is a set of bags over elements in X,
$\delta_{ext}(s, e, \phi) = (s, e)$.
$\delta_{con} : S \times X^b \rightarrow S$: confluent transition function.
$\lambda : S \rightarrow Y^b$: output function.
$ta : S \rightarrow R_{0+\rightarrow\infty}$: time advance function,
where $Q = \{(s, e) | s \in S, 0 < e < ta(s)\}$,
e is the elapsed time since last state transition.

δ_{con} is the difference from the previous *DEVS* formalisms. It gives the modeler complete control over the collision behavior when a component receives events at the time of its internal transition, $e = 0$ or $e = ta(s)$. Rather than serializing model behavior at collision times, the *P-DEVS* formalism leaves this decision of what serialization to use, if any, to the modeler. Indeed, if so desired, the *E-DEVS* formalism can be recovered by setting $\delta_{con}(s, x^b)$ to $\delta_{ext}(s_n, 0, x_n)$, where $n \geq 1$, $s_1 = \delta_{int}(s)$, $s_n = \delta_{ext}(s_{n-1}, 0, x_{n-1})$ when $n > 1$, and x_n is a desired serialization defined by $Order(x^b)$.

The semantics of the *Parallel DEVS* are illustrated in *Figure 2*. The internal transitions are carried out at the next event time for all imminent components receiving no external events. Also, external events generated by these imminents trigger external transitions at receptive non-imminents (those components for which there are no internal transitions scheduled at the event receiving time). However, for those components for which the internal and external transitions collide, the *confluent transition function* is employed instead of either the internal or external transition function to determine the new state. In *Figure 2*, all the state transitions are the same as those in the original *DEVS* formalism except at $t2$ and $t5$ where collisions occur. At these instances, the new states are defined to be $s_3 = \delta_{con}(s_2, x_2)$ and $s_6 = \delta_{con}(s_5, x_5)$. We still need the *bag*, x^b, to collect simultaneous external events generated by internal and confluent transitions.

The structure of the revised *coupled model* is —

$$DN =< X, Y, D, \{M_i\}, \{I_i\}, \{Z_{i,j}\} >$$

X: a set of input events.
Y: a set of output events.
D: a set of components.
for each i in D,
M_i is a component.
for each i in $D \cup \{self\}$, I_i is the influencees of i.
for each j in I_i,
$Z_{i,j}$ is a function,
the i-to-j output translation.

The structure is subject to the constraints that for each i in D,
$M_i =< X_i, S_i, Y_i, \delta_{inti}, \delta_{exti}, \delta_{coni}, ta_i >$ is a *P-DEVS* structure,
I_i is a subset of $D \cup \{self\}$, i is not in I_i,
$Z_{self,j} : X_{self} \rightarrow X_j$,
$Z_{i,self} : Y_i \rightarrow Y_{self}$,
$Z_{i,j} : Y_i \rightarrow X_j$.

Here *self* refers to the coupled model itself and is a device for allowing specification of external input and external output couplings.

4 CLOSURE UNDER COUPLING

The demonstration of the closure is done by constructing the resultant of a coupled model and showing it to be a well defined *P-DEVS*. The *resultant* of a coupled model ($DN =< X, Y, D, \{M_i\}, \{I_i\}, \{Z_{i,j}\} >$) is a *P-DEVS* model ($M =< X, S, Y, \delta_{int}, \delta_{ext}, \delta_{con}, \lambda, ta >$), where
$S = \times Q_i$ where $i \in D$.
$ta(s) = \text{minimum}\{\sigma_i | i \in D\}$,
where $s \in S$ and $\sigma_i = ta(s_i) - e_i$.
Let
$s = (..., (s_i, e_i), ...)$,
$IMM(s) = \{i | \sigma_i = ta(s)\}$,
$INF(s) = \{j | j \in \cup_{i \in IMM(s)} I_i\}$,
$CONF(s) = IMM(s) \cap INF(s)$,

$INT(s) = IMM(s) - INF(s)$,
$EXT(s) = INF(s) - IMM(s)$.

We partition the components into four sets at any transition time. $INT(s)$ contains the components ready to make an internal transition without input events. $EXT(s)$ contains the components receiving input events but not scheduled for an internal transition. $CONF(s)$ contains the components receiving input events and also scheduled for internal transitions at the same time. $UN(s)$ contains the remaining components. Then,
$\lambda(s) = \{Z_{i,self}(\lambda_i(s_i)) | i \in IMM(s) \wedge self \in I_i\}$.
$\delta_{int}(s) = (\ldots, (s'_i, e'_i), \ldots)$,
where
$(s'_i, e'_i) = (\delta_{inti}(s_i), 0)$ for $i \in INT(s)$,
$(s'_i, e'_i) = (\delta_{exti}(s_i, e_i + ta(s), x^b_i), 0)$ for $i \in EXT(s)$,
$(s'_i, e'_i) = (\delta_{coni}(s_i, x^b_i), 0)$ for $i \in CONF(s)$,
$(s'_i, e'_i) = (s_i, e_i + ta(s))$ otherwise $i \in UN(s)$,
and
$x^b_i = \{Z_{o,i}(\lambda_o(s_o)) | o \in IMM(s) \wedge i \in I_o\}$.

The resultant internal transition comprises of four kinds of component transitions: internal transitions of $INT(s)$ components, external transitions of $EXT(s)$ components, confluent transitions of $CONF(s)$ components and the remainder,$UN(s)$, whose elapsed times are merely updated by $ta(s)$. (The participation of $UN(s)$ can be removed in simulation by using an absolute time base rather than the relative elapsed time.)

Note that by assumption, this is an internal transition of the resultant model, and there is no external event being received by the coupled model at this time.

Next, we construct the δ_{ext} of the resultant.
$\delta_{ext}(s, e, x^b) = (\ldots, (s'_i, e'_i), \ldots)$,
where
$(s'_i, e'_i) = (\delta_{exti}(s_i, e_i + e, x^b_i), 0)$ for $i \in I_{self}$,
$(s'_i, e'_i) = (s_i, e_i + e)$ otherwise,
and
$x^b_i = \{Z_{self,i}(x) | x \in x^b \wedge i \in I_{self}\}$.

The incoming event bag, x^b is translated and routed to the event bag, x^b_j, of each influenced child, j. The resultant's external transition comprises all the external transitions of the influenced children.

Finally, we construct the δ_{con} of the resultant. Let
$INF'(s) = \{j | j \in \cup_{i \in (IMM(s) \cup \{self\})} I_i\}$,
$CONF'(s) = IMM(s) \cap INF'(s)$,
$INT'(s) = IMM(s) - INF'(s)$,
$EXT'(s) = INF'(s) - IMM(s)$.
$\delta_{con}(s, x^b) = (\ldots, (s'_i, e'_i), \ldots)$,
where
$(s'_i, e'_i) = (\delta_{inti}(s_i), 0)$ for $i \in INT'(s)$,
$(s'_i, e'_i) = (\delta_{exti}(s_i, e_i + ta(s), x^b_i), 0)$ for $i \in EXT'(s)$,
$(s'_i, e'_i) = (\delta_{coni}(s_i, x^b_i), 0)$ for $i \in CONF'(s)$,
$(s'_i, e'_i) = (s_i, e_i + ta(s))$ otherwise,
and
$x^b_i = \{Z_{o,i}(\lambda_o(s_o)) | o \in IMM(s) \wedge i \in I_o\} \uplus \{Z_{self,i}(x) | x \in x^b \wedge i \in I_{self}\}$.

To establish closure under coupling, we must also consider how to define the δ_{con} of the resultant, i.e., what happens to the coupled model when some of the children are scheduled to make internal transitions and are also about to receive external events originating outside the boundaries of the model. Fortunately, it turns out that the difference between δ_{con} of the resultant and its δ_{int} is simply the extra confluent effect produced by the incoming event bag, x^b, at simulation time $ta(s)$. By redefining the influencee set to $INF'(s)$ that includes the additional influencees from the incoming couplings, $z(self, i)$, we come up with three similar groups for δ_{con}. The hierarchical consistency is achieved here by the $\uplus$ operation that gathers all external events, whether internally or externally generated, at the same time into one single event group.

Examining the application of the three transition functions, we see that the semantics of the applications of the three transition functions are maintained throughout the hierarchical composition. $IMM(s)$ identifies components with $e_i = ta(s)$. Components that are not in $IMM(s)$ have elapsed times, $0 \leq e_l < ta(s)$. $INF(s)$ identifies components with $x^b_i \neq \phi$ (the empty bag). Combining $IMM(s)$ and $INF(s)$, we can identify which function should be applied to which component easily. For example, δ_{con} is strictly applied to elements in $CONF(s) = IMM(s) \cap INF(s)$ which only contains components with $e_i = ta(s)$ and $x^b_i \neq \phi$.

From the definition of the δ_{int}, δ_{con}, and δ_{ext}, we see that they are special cases of a more generic transition function $\delta(s, e, x^b)$(Zeigler 1984). δ_{int} is applied to the cases when $(s, e, x^b) = (s, ta(s), \phi)$, δ_{con} to the cases when $(s, e, x^b) = (s, ta(s), x^b)$ where $x^b \neq \phi$, and, δ_{ext} to (s, e, x^b) where $0 \leq e < ta(s)$ and $x^b \neq \phi$.

Because of the distinctive semantics among the three transition functions at the atomic model level, it is advantageous for a modeler to deal with three different semantics individually rather than lumping them into one generic transition function. However, for the resultant model, and transparent to the modeler, the three transition functions can be merged into a more generic transition function so that the same

implementation can be reused by all transitions.
$\delta_{con}(s, x^b) \equiv \delta(s, ta(s), x^b)$,
$\delta_{ext}(s, e, x^b) \equiv \delta(s, e, x^b)$ for $0 < e < ta(s)$, and
$\delta_{int}(s) \equiv \delta(s, ta(s), \phi)$,

To summarize, here is the generic transition function.
$\delta(s, e, x^b) = (..., (s'_i, e'_i), ...)$,
where
$(s'_i, e'_i) = (\delta_i(s_i, ta(s), \phi), 0)$ where $i \in INT(s)$,
$(s'_i, e'_i) = (\delta_i(s_i, e_i + e, x^b_i), 0)$ where $i \in EXT(s)$,
$(s'_i, e'_i) = (\delta_i(s_i, ta(s), x^b_i), 0)$ where $i \in CONF(s)$,
$(s'_i, e'_i) = (s_i, e_i + e)$ otherwise,
and
$$x^b_i = \{Z_{o,i}(\lambda_o(s_o)) | o \in IMM(s) \wedge i \in I_o\} \uplus \{Z_{self,i}(x) | x \in x^b \wedge i \in I_{self}\}.$$

A single level abstract simulator has been prototyped on the CM-5 massively parallel computer (Chow and Zeigler 1994).

5 CONCLUSIONS

The state that the external transitions and output functions see must be uniquely determined for a model at any given time. The original *DEVS* formalism provides a way to uniquely determine states but sometimes in a non-intuitive manner because of the serial nature of its abstract simulator. *Extended DEVS* formalism helps modelers define the unique state by first carrying out internal transitions and than external transitions at the same simulation time. Though this approach renders more intuitively asserted models and exposes parallelism, it severely limits the latitude of control over the model's collision behaviors.

In the *Parallel DEVS* formalism, a modeler is required to supply the additional confluent transition function that captures the collision behavior. This function allows the coupling construction to follow the semantics of a collision down to the atomic level. Closure under coupling and hierarchical consistency follow. In addition, this property allows a deep hierarchical model to be *flattened* to a single level. This restructuring can support more efficient simulation, particularly in sequential environments.

The *P-DEVS* formalism also supports the bag concept for simultaneous events so that modelers can put more effort into the design of external transition functions and combine executions of several external transitions into a single one. Not only does this create more intuitive and correct simulation results, it also speeds up simulation when many external events occur at the same time for a given model. The well isolated transition groups at each coupling level add to the existing possibilities to exploit the parallelism of the hierarchical *DEVS* models.

Finally, the *Parallel DEVS* provides a sound framework for language developers to experiment with new forms of discrete event model expression so as to render the abstract specifications in concrete form. For example, many language constructs could be developed for the specification of the confluent transition function. The logic-based formalism of Radiya and Sargent (1994) is suggestive in this context.

ACKNOWLEDGMENTS

This research was partially supported by NSF HPCC Grand Challenge Application Group Grant ASC-9318169 with ARPA participation and employed the CM-5 at NCSA under grant MCA94P02

REFERENCES

Chow, A. C., and B. P. Zeigler. 1994. The simulators of the Parallel DEVS formalism. *Proceedings of the Fifth Annual Conference on AI, Simulation and Planning in High Autonomy Systems*, not final.

Fujimoto, R. M. 1990. Parallel discrete event simulation. *Communications of the ACM*, 33(10):30–53.

Kim, T. G. 1994. DEVSIM++ User's Manual, CORE Lab, EE Dept, KAIST, Taejon, Korea.

Lubachevesky, B., A. Weiss, and A. Schwartz. 1991. An analysis of rollback-based simulation. *ACM Transactions on Modeling and Computer Simulation*, 1(2).

Pegden, C. D., and D. A. Davis. 1992. Arena TM : A SIMAN/CINEMA based hierarchical modeling system. *Winter Simulation Conference Proceedings*, Phoenix, AZ.

Praehofer, H. 1993. An environment for DEVS-based multiformalism simulation in common Lisp/CLOS. *Discrete Event Dynamic Systems*, 3.

Preiss, B. 1989. The Yaddes distributed discrete event simulation specification language and execution environment. *Distributed Simulation 89*. SCS Press.

Radiya, A., and R. G. Sargent. 1994. A logic-based foundation of discrete event modeling and simulation. *ACM Transactions on Modeling and Computer Simulation*, 4(1).

Rozenblit, J. W., and P. Janknski. 1990. An integrated framework for knowledge-based modeling and simulation of natural systems. *Simulation*, 57(3).

Sargent, R. 1993. Hierarchical modeling for discrete event simulation (panel). *Winter Simulation Con-*

ference Proceedings, page 569, Los Angeles, CA.

Wang, Y. H. 1992. *Discrete-event simulation on a massively parallel computer*, Ph. D. dissertation, Dept. of Electrical and Computer Engineering, University of Arizona, Tucson, AZ.

Wang, Y. H., and B. P. Zeigler. 1992. Extending the DEVS Formalism for Massively Parallel Simulation. *Discrete Event Dynamic Systems: Theory and Applications*, 3:193–218.

Zeigler, B. P. 1976. *Theory of Modelling and Simulation*. Wiley-Interscience, New York, 1976.

Zeigler, B. P. 1984. *Multifacetted Modelling and Discrete Event Simulation*. Academic Press, London.

Zeigler, B. P. 1990. *Object-Oriented Simulation with Hierarchical, Modular Models*. Academic Press, San Diego, California.

AUTHOR BIOGRAPHIES

ALEX CHUNGHEN CHOW has joined IBM Corporation since 1991. He received his B.S. and M.S. degrees in Electrical Engineering from National Chiao-Tung University in Taiwan, and his Ph. D. degree in Electrical Engineering from the University of Arizona in 1990. His research interests include visual programming environment, system modeling and simulation, graphical user interface, and object-oriented programming framework.

BERNARD P. ZEIGLER is a professor of Electrical and Computer Engineering at the University of Arizona, Tucson. He received his B.S. Eng. Phys. from McGill University, 1962, M.S.E.E. from MIT, 1964, and Ph. D. from the University of Michigan in 1969. He has published over two hundred journal and conference articles in modelling and simulation, knowledge based systems and high autonomy systems. His first book "Theory of Modelling and Simulation" (Wiley,1976) is regarded as one of the foundational works in the field. A second book "Multifacetted Modelling and Discrete Event Simulation" (Academic Press, 1984), was given the outstanding simulation publication award Concepts developed in earlier work are implemented in the DEVS simulation environment and applied to high autonomy issues in the latest book, "Object-oriented Simulation with Hierarchical, Modular Models: Intelligent Agents and Endomorphic Systems" published by Academic Press, Boston, 1990. Zeigler's research has been supported by federal agencies including NSF, NASA, USAF, and the US Army, as well as industrial sponsors including Siemens, McDonnell Douglas, and Motorola. He is currently heading a multidisciplinary team to demonstrate an innovative approach to massively parallel simulation of large scale ecosystem models within NSF's HPCC Grand Challenge initiative. He was elected as Fellow of the IEEE for his innovative work in discrete event modelling theory.