Proceedings of the 1995 Winter Simulation Conference
ed. C. Alexopoulos, K. Kang, W. R. Lilegdon, and D. Goldsman

DYNAMIC STRUCTURE DISCRETE EVENT SYSTEM SPECIFICATION: A NEW FORMALISM FOR DYNAMIC STRUCTURE MODELING AND SIMULATION

Fernando J. Barros

Departamento de Engenharia Informática
Universidade de Coimbra, Pólo II
P-3030 Coimbra, PORTUGAL

ABSTRACT

Traditional simulation methodologies do not support changes in model structure during a simulation run. Current methodologies support only changes in model descriptive variables. Changes in structure are thus forced to be represented at the simple behavioral level. Many models are better represented at both behavioral and structural level. We present a new simulation formalism that full supports changes in model structure and its closure under coupling. The Dynamic Structure Discrete Event System Specification formalism (DSDEVS) supports a new simulation paradigm, structural simulation, as opposed to conventional trajectory simulation. This new formalism supports changes in structure to the full extent, ranging from simple model/connection add/deletion to the exchange of models between network of models. The DELTA simulation environment, an implementation of the DSDEVS formalism, is briefly described.

1 INTRODUCTION

Conventional simulation formalisms support a very limited description of dynamic models. They can only represent changes of the variables describing the state of the models, i.e., behavioral changes. Changes in model structure must thus be mapped onto changes of model descriptive variables. Many kinds of models are better represented by both structural and behavioral changes. Examples of these systems include adaptive computer architectures (Zeigler, Kim and Lee 1991), fault tolerance computers and self adaptive systems. A new simulation formalism is needed to handle changes in structure in the same way that conventional formalisms can handle changes in descriptive variables. Previous work related to dynamic structure modeling and simulation can be found in Zeigler and Praehofer (1989), Barros, Mendes and Zeigler (1994), Uhrmacher and Arnold (1994), and Thomas (1994).

In this paper we define a new simulation modeling formalism, *Dynamic Structure Discrete Event System Specification (DSDEVS)*. The DSDEVS formalism, based on the DEVS formalism (Zeigler 1976, Zeigler 1984, Zeigler 1990), provides full support to dynamic structure modeling and simulation. The DEVS formalism provides a formal representation of discrete event dynamic systems. Hierarchy and modularity are very important features for modeling and simulation of very complex systems. These features have been incorporated in some simulation environments (Zeigler 1990, Barros and Mendes 1993).

The DSDEVS formalism supports changes in structure by the introduction of a special model, named here by *network executive*, that keeps in its internal state the structure of a network of models. Changes in executive state are automatically mapped into changes in structure. Because each network has a corresponding executive, changes in structure can be made at any level of model hierarchy. Changes in model structure are full supported, ranging from simples changes in model interconnection to exchanges of models between networks.

More details about the DSDEVS formalism and its abstract simulators can be found in Barros (1995).

2 THE DSDEVS FORMALISM

The Dynamic Structure Discrete Event System Specification formalism (DSDEVS) is a new formalism to specify system that can change their structure dynamically. In the DSDEVS formalism there are two types of models, basic models and network models.

2.1 The DSDEVS Basic Model

Basic models are defined in the DSDEVS formalism, as in the DEVS formalism, by the structure

$$M = \langle X,S,Y,\delta_{int},\delta_{ext},\lambda,\tau \rangle$$

where

$X \equiv$ set of input events,

$S \equiv$ set of sequential states,

$Y \equiv$ set of output events,

δ_{int}: $S \rightarrow S \equiv$ internal transition function,

δ_{ext}: $X \times Q \rightarrow S \equiv$ external transition function, where

$Q = \{(s,e) \mid s \in S, 0 \leq e \leq \tau(s)\} \equiv$ total state set,

$e \equiv$ time elapsed since last transition,

λ: $S \rightarrow Y \equiv$ output function,

τ: $S \rightarrow \mathbf{R}_0^+ \equiv$ time advance function.

2.2 The DSDEVS Network Model

Network models are a combination of the DSDEVS basic models. In opposition to other simulation formalisms, that supports only static structure models, the structure of the DSDEVS networks can be changed. The DSDEVS *dynamic structure network* is defined by the structure

$$DSDEVN = \langle \chi, M_\chi \rangle$$

where

$\chi \equiv$ DSDEVS *network executive*,

$M_\chi \equiv$ model of χ.

The DSDEVS network is defined with a special component, the *network executive* χ. M_χ, the model of the executive, is a DSDEVS basic model and is defined by the structure

$$M_\chi = \langle X_\chi,S_\chi,Y_\chi,\delta_{int_\chi},\delta_{ext_\chi},\lambda_\chi,\tau_\chi \rangle.$$

The information about the dynamic structure network is located in the state of the executive. The executive, besides domain dependent state knowledge, has information about network composition and coupling. Changes in executive network related state variables will be automatically mapped onto changes in network structure.

The state $s_\chi \in S_\chi$ has information about the structure of the DSDEVS network and is defined by the tuple

$$s_\chi = (X_\Delta,Y_\Delta,D,\{M_i\},\{I_i\},\{Z_{i,j}\},\Xi,\theta)$$

where

$X_\Delta \equiv$ input event set of the DSDEVS network, with

$\Delta \equiv$ DSDEVS dynamic structure network,

$Y_\Delta \equiv$ output event set of the DSDEVS network,

$D \equiv$ set of components,

$M_i \equiv$ model of component i, for all $i \in D$,

$I_i \equiv$ influencees of i, for all $i \in D \cup \{\chi,\Delta\}$,

$Z_{i,j} \equiv i$-to-j translation function, for all $j \in I_i$,

$\Xi \equiv$ select function,

$\theta \equiv$ other state variables not defined before.

The state variables are subject to the following constraints:

$\chi \notin D$,

$M_i = \langle X_i,S_i,Y_i,\delta_{int_i},\delta_{ext_i},\lambda_i,\tau_i \rangle \equiv$ DSDEVS basic model,

for all $i \in D$,

$i \notin I_i$, for all $i \in D \cup \{\chi,\Delta\}$,

$\Xi : \Pi \rightarrow D \cup \{\chi\}$, where

$\Pi = 2^{D \cup \{\chi\}} - \{\}$,

$\Xi(A) \in A$,

$Z_{\Delta,j} : X_\Delta \rightarrow X_j$,

$Z_{i,\Delta} : Y_i \rightarrow Y_\Delta$,

$Z_{i,j} : Y_i \rightarrow X_j$,

$Z_{k,\chi}(y) \neq \varnothing \Rightarrow Z_{k,j}(y) = \varnothing$,

for $k \in D \cup \{\Delta\}$ and for all $j \in I_k - \{\chi\}$, with

$\varnothing \equiv$ null event.

The last constraint states that if a model sends an external event to the executive, then the executive is the only component receiving the event. This constraint is added to prevent ambiguity that would arise when several components, including the executive, receive an input. In this case the behavior of the system would depend on the external event handling order, due to the possible change in the structure made by the executive.

In this discussion we have limited ourselves to a sequential execution. A discussion of a parallel execution can be found in Chow and Zeigler (1994).

As the coupling information of the network is located in the state of the executive, transition functions can change this state and by consequence change the structure of the network. Usually the state variables represented by θ are used to keep information about when and how changes take place.

The tuple $(X_\Delta,Y_\Delta,D,\{M_i\},\{I_i\},\{Z_{i,j}\},\Xi)$ defined in the executive state is referred to here as the *network*

structure. Any change in one of these variables is called a *change in structure*. Here changes in structure are defined in a broad sense, ranging from the simple change of the select function to more complex changes like the addition/deletion of components.

3 THE DSDEVS CLOSURE UNDER COUPLING

A formalism is closed under coupling if any network of models specified by the formalism can be itself specified by the formalism (Zeigler 1984). To use the DSDEVS formalism for modeling large simulation models, one must ensure that it supports model building in a hierarchical and modular manner. A DSDEVS model must obey the rules stated in Zeigler (1990) for hierarchical model construction:

- A basic model is a hierarchical model
- A network model whose components are hierarchical models is a hierarchical model
- Nothing else is a hierarchical model

To build models in a hierarchical manner, we need to prove that a DSDEVS network can be itself expressed as a DSDEVS model. Then we can build models recursively with any arbitrary levels of hierarchy. The equivalence between a DSDEVS network and a DSDEVS basic model is called the closure under coupling property of the DSDEVS formalism. To prove that the DSDEVS formalism is closed under coupling, it is necessary to obtain the equivalent DSDEVS basic model from the dynamic structure network model ($DSDEVN = \langle \chi, M_\chi \rangle$). This is illustrated by the description of all the elements present in the basic model structure ($M = \langle X, S, Y, \delta_{int}, \delta_{ext}, \lambda, \tau \rangle$):

$X = X_{\Delta,\chi}$, where
$X_{\Delta,\chi}$ represents the input event set of the DSDEVS network defined in the state of χ,

$S = \times Q_i$ for all $i \in D_\chi \cup \{\chi\}$,

$Y = Y_{\Delta,\chi}$,

$\tau(s) = \min\{\sigma_i \mid i \in D_\chi \cup \{\chi\}\}$, $s \in S$, where
$\sigma_i = \tau_i(s_i) - e_i$,

$\lambda(s) = Z_{i^*,\Delta,\chi}(\lambda_{i^*}(s_{i^*}))$, $\Delta \in I_{i^*,\chi}$, $s \in S$, with
$i^* = \Xi_\chi(\{i \mid i \in D_\chi \cup \{\chi\}, \sigma_i = \tau(s)\})$.

To describe the remaining equivalent functions δ_{ext} and δ_{int}, we divide the components of the network in three sets. The first set has just the executive, $\{\chi\}$. In the second set $R = \{r_1,\ldots,r_n\}$, we include the models that will be removed by the executive in response to an internal or external transition. This set is the empty set if no model is removed. The third set is composed by the remaining components $\{\ldots,i,\ldots\}$, where i represents an arbitrary component. The state $s \in S$ is thus defined by

$$s = ((s_\chi,e_\chi),\ldots,(s_i,e_i),\ldots,(s_{r_1},e_{r_1}),\ldots,(s_{r_n},e_{r_n}))$$

where $s_\chi = (X_\Delta, Y_\Delta, D, \{M_i\}, \{I_i\}, \{Z_{i,j}\}, \Xi, \theta)$ represents the state of the network executive. By the definition of the state S, when components are added or removed, the state $s \in S$ changes accordingly, as we are going to demonstrate.

$\delta_{ext}(x,s,e)$ is defined by

$\delta_{ext}(x,s,e) = ((s_\chi, e_\chi + e),\ldots,(s'_i,0),\ldots)$ for all $i \in I_{\Delta,\chi}$,

where

$s'_i = \delta_{ext_i}(Z_{\Delta,i,\chi}(x), s_i, e_i + e)$ with

$Z_{\Delta,i,\chi}(x) \neq \varnothing$,

$R = \{\}$ ≡ removed components,

or by

$\delta_{ext}(x,s,e) = ((s'_\chi,0),\ldots,(s_i,e_i + e),\ldots,(s_{a_1},0),\ldots,(s_{a_k},0))$

if $\chi \in I_{\Delta,\chi}$, where

$s'_\chi = \delta_{ext_\chi}(Z_{\Delta,\chi,\chi}(x),$

$(X_\Delta, Y_\Delta, D, \{M_i\}, \{I_i\}, \{Z_{i,j}\}, \Xi, \theta), e_\chi + e) =$

$(X'_\Delta, Y'_\Delta, D', \{M'_i\}, \{I'_i\}, \{Z'_{i,j}\}, \Xi', \theta')$, with

$Z_{\Delta,\chi,\chi}(x) \neq \varnothing$,

$D' = D - R \cup A$ with

$A = \{a_1,\ldots,a_k\}$ ≡ added components,

$R = \{r_1,\ldots,r_n\}$ ≡ removed components.

State variables X'_Δ, Y'_Δ are the new set of network input and output events respectively. State variables $\{M'_i\}$, $\{I'_i\}$, $\{Z'_{i,j}\}$ and Ξ' are now defined over D' instead of D.

By the constraints existing on $Z_{i,j}$, if χ receives an external event, no other model will receive external events. Thus only one branch of the function will be used at each time.

$\delta_{int}(s) = \delta^*_{ext}(i^*, y^*, \delta^*_{int}(i^*, s))$, where

$i^* = \Xi_\chi(\{i \mid i \in D_\chi \cup \{\chi\}, \tau_i(s_i) - e_i = \tau(s)\})$,

is the imminent component, and

$y^* = \lambda_{i^*}(s_{i^*})$

is the output of i^*.

The function δ^*_{int} is used to compute the effect of the internal transition of the imminent component. This function is expressed by

δ^*_{int}: $D^* \times S \rightarrow S$ where $D^* = D_\chi \cup \{\chi\}$,

and defined by

$\delta^*_{int}(i^*,s) = ((s_\chi, e_\chi + \tau(s)),\ldots,(\delta_{int_{i^*}}(s_{i^*}),0),\ldots)$ if $i = i^*$,

with $i \neq \chi$, $R = \{\}$,

or by

$\delta^*_{int}(i^*,s) = ((s'_\chi,0),\ldots,(s_i,e_i + \tau(s)),\ldots,(s_{a_1},0),\ldots,(s_{a_k},0))$

if $\chi = i^*$, where

$s'_\chi = \delta_{int_\chi}((X_\Delta,Y_\Delta,D,\{M_i\},\{I_i\},\{Z_{ij}\},\Xi,\theta))$

$= (X'_\Delta,Y'_\Delta,D',\{M'_i\},\{I'_i\},\{Z'_{ij}\},\Xi',\theta')$, with

$D' = D - \{r_1,\ldots,r_n\} \cup \{a_1,\ldots,a_k\}$.

State variables $\{M'_i\}$, $\{I'_i\}$, $\{Z'_{ij}\}$ and Ξ' are now defined over D' instead of D.

The function δ^*_{ext} is used to compute the effect of the output of the imminent component on the equivalent total state set. This function is expressed by

δ^*_{ext}: $D^* \times Y^* \times S \rightarrow S$ where $Y^* = \cup_{i \in D^*} Y_i$,

and defined by

$\delta^*_{ext}(i^*,y^*,s) = ((s_\chi, e_\chi + \tau(s)),\ldots,(s'_i,0)\ldots)$ for all $i \in I_{i^*,\chi}$,

where

$s'_i = \delta_{ext_i}(Z_{i^*,i,\chi}(y^*),s_i,e_i + \tau(s))$, with

$Z_{i^*,i,\chi}(y^*) \neq \varnothing$, $R = \{\}$,

or by

$\delta^*_{ext}(i^*,y^*,s) = ((s'_\chi,0),\ldots,(s_i,e_i + \tau(s)),\ldots,$

$(s_{a_1},0),\ldots,(s_{a_k},0))$ if $\chi \in I_{i^*,\chi}$,

where

$s'_\chi = \delta_{ext_\chi}(x,(X_\Delta,Y_\Delta,D,\{M_i\},\{I_i\},\{Z_{ij}\},\Xi,\theta),e_\chi+\tau(s))$

$= (X'_\Delta,Y'_\Delta,D',\{M'_i\},\{I'_i\},\{Z'_{ij}\},\Xi',\theta')$, with

$x = Z_{i^*,\chi,\chi}(y^*)$, $x \neq \varnothing$,

$D' = D - \{r_1,\ldots,r_n\} \cup \{a_1,\ldots,a_k\}$.

State variables $\{M'_i\}$, $\{I'_i\}$, $\{Z'_{ij}\}$ and Ξ' are now defined over D' instead of D.

As seen by the description of the equivalent transition functions δ_{ext} and δ_{int}, the network structure can change when the executive performs an internal or an external transition. As all structure information is kept in executive state, any change in structure can be accomplished by the executive transition functions. DSDEVS provides thus a formalism to represent and simulate a new class of models with a time-varying structure. Although we have proved that any dynamic structure network model could be reduced to a basic model and thus be simulated without any change in structure, this transformation will be prohibited by model complexity in real situations. From a practical point of view it will be very difficult or even impossible to simulate an all range of complex dynamic structure models without the help of a formalism with sound structural semantics.

4 THE DELTA ENVIRONMENT

The DELTA modeling and simulation environment is an implementation of the DSDEVS formalism and provides full support for modeling and simulation of dynamic structure discrete event systems. DELTA has been implemented in the Smalltalk/V language (Digitalk 1992), and supports a class based realization of the DSDEVS formalism. The class hierarchy for the DELTA modeling simulation environment is represented in Figure 1. Class **DSDEVSObject** is the parent class for the classes **Models** and **Simulators**. **BasicModel** class supports DSDEVS basic models and **NetworkModel** class supports DSDEVS networks. The **Executive** class is an abstract class and domain dependent executives can be obtained by subclassing this class. The subclasses of **Simulators** implement the abstract simulators necessary to interpret models implicit behavior. A description of the abstract simulators is given in Barros (1995).

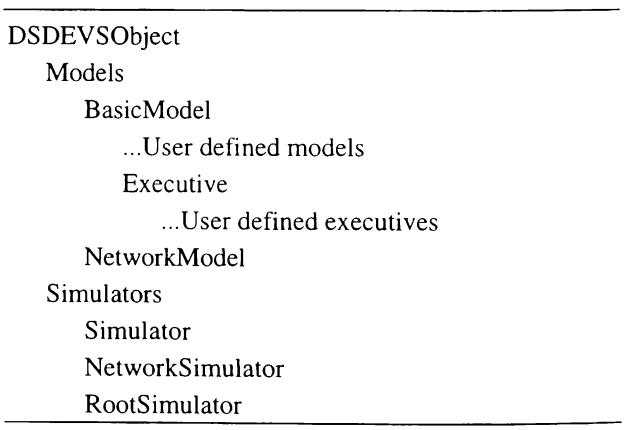

Figure 1: Class Hierarchy in DELTA

To support changes in structure the following methods were defined in the executive class:

addModel: *aModel*, adds a new model to the simulation;

removeModel: *aModel*, removes a model from simulation. All links from and to the model are removed;

replace: *aModel* **with**: *bModel*, replaces one model by a new model. Links remain the same;
link: *aModel* **port**: *aPort* **to**: *bModel* **port**: *bPort*, creates a link between two models;
unlink: *aModel* **port**: *aPort* **to**: *bModel* **port**: *bPort*, deletes a link between two models;
unlink: *aModel* **port**: *aPort*, remove all links from a specific port of a model;
find: *aName*, find the model with a *aName*;
clear, removes all the models except the executive;
models, returns a list of the models in the network.

These methods can be used by the executive element to change the structure of the network it is controlling. Because each DSDEVS network has its own executive, changes in structure can also be made at any level of the hierarchy.

Due to space limitations it was not possible to include an application example. More details about applications are given in Barros (1995).

5 CONCLUSIONS

We have described the Dynamic Structure Discrete Event System Specification (DSDEVS), a new simulation formalism for modeling, and its closure under coupling. The DSDEVS formalism provides complete and sound structural semantics. It proved to be a powerful modeling tool for representing a wide range of discrete event systems that are more easily expressed as dynamic structure models. DSDEVS also allows models to be constructed in a modular and hierarchical manner. Structural changes are supported in a full extent and changes can be made at any level of the hierarchy. The DELTA modeling and simulation environment, a full implementation of the DSDEVS formalism, was briefly described.

ACKNOWLEDGMENTS

The author thanks Professor Bernard P. Zeigler of the University of Arizona, who introduced him to Variable Structure Modeling, his valuable suggestions. This work was supported by a grant from *COMISSÃO INVOTAN*.

REFERENCES

Barros, F. J., and M. T. 1993. Modelling and Simulation of an Integrated Circuit Flow-Shop in DEVS-V. *Proceedings of the European Simulation Symposium*. SCS Publications, 103-108.

Barros, F. J., M. T. Mendes, and B. P. Zeigler. 1994. Variable DEVS - Variable Structure Modeling Formalism: An Adaptive Computer Architecture Example. *Proceedings of the Fifth Annual Conference on AI, Simulation and Planning in High Autonomy Systems*. IEEE Computer Press, 185-191.

Barros, F. J. 1995. Dynamic Structure Discrete Event System Specification: Formalism and Abstract Simulators. Technical Report DEE-FCTUC-001-95, Department of Electrical Engineering, University of Coimbra, Portugal.

Chow, A. C., and B. P. Zeigler. 1994. Abstract Simulator for the Parallel DEVS Formalism. *Proceedings of the Fifth Annual Conference on AI, Simulation and Planning in High Autonomy Systems*. IEEE Computer Press, 157-163.

Digitalk. 1992. *Smalltalk for Windows: Tutorial and Programming Handbook*. Los Angeles: Digitalk Inc.

Thomas, C. 1994. Interface-Oriented Classification of DEVS Models. *Proceedings of the Fifth Annual Conference on AI, Simulation and Planning in High Autonomy Systems*. IEEE Computer Press, 208-213.

Uhrmacher, A. M., and R. Arnold. 1994. Distributing and Maintaining Knowledge: Agents in Variable Structure Environment. *Proceedings of the Fifth Annual Conference on AI, Simulation and Planning in High Autonomy Systems*. IEEE Computer Press, 178-184.

Zeigler, B. P. 1976. *Theory of Modelling and Simulation*. New York: John Wiley.

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

Zeigler, B. P., and H. Praehofer. 1989. Systems Theory Challenges in the Simulation of Variable Structure and Intelligent Systems. In *CAST-Computer-Aided Systems Theory*, eds. F. Pichler and Moreno-Diaz, 41-51. Berlin: Springer Verlag.

Zeigler, B. P. 1990. *Object-Oriented Simulation with Hierarchical, Modular Models*. New York: Academic Press.

Zeigler, B. P., T. G. Kim, and C. Lee. 1991. Variable Structure Modeling Methodology: An Adaptive Computer Architecture Example. *Transactions of the Society for Computer Simulation* 7:291-319.

AUTHOR BIOGRAPHY

FERNANDO J. BARROS is a Ph.D. candidate in the Department of Computer Engineering at the University of Coimbra, Portugal. He received his M.S. degree from the same university in 1992. His research interests include Simulation Methodology, Manufacturing Systems and Self-Adaptive Systems. He his a member of SCS, ACM, IEEE and IIE.