Modeling Formalisms for Dynamic Structure Systems

FERNANDO J. BARROS
Universidade de Coimbra

We present a new concept for a system network to represent systems that are able to undergo structural change. Change in structure is defined in general terms, and includes the addition and deletion of systems and the modification of the relations among components. The structure of a system network is stored in the network executive. Any change in structure-related information is mapped into modifications in the network structure.

Based on these concepts, we derive three new system specifications that provide a shorthand notation to specify classes of dynamic structure systems. These new formalisms are: dynamic structure discrete time system, dynamic structure differential equation specified systems, and dynamic structure discrete event system specification. We demonstrate that these formalisms are closed under coupling, making hierarchical model construction possible. Formalisms are described using set theoretic notation and general systems theory concepts.

Categories and Subject Descriptors: I.6.1 [**Simulation and Modeling**]: Modeling Theory

General Terms: Design, Experimentation, Performance, Theory

Additional Key Words and Phrases: Dynamic structure systems specifications, parallel modeling formalisms.

1. INTRODUCTION

General systems theory provides a formal representation of dynamic systems. The book [Zeigler 1976] is a milestone in modeling and simulation theory. It offers modeling formalisms for the specification of the most common dynamic systems, like the DEVS formalism to model discrete event systems. General systems theory, however, was developed to represent systems with a time invariant structure [Mesarovic and Takahara 1975; Wymore 1977; Zeigler 1976; Zeigler 1985]. We describe a new approach to the representation of systems that can undergo structural changes. These systems are referred to here as *dynamic structure systems.* The concept of dynamic structure systems network is formalized and the dynamic structure network system specifications for the three more com-

Author's address: Universidade de Coimbra, Dep. de Engenharia Informática, Pólo II, P-3030 Coimbra, Portugal. Email: ⟨barros@dei.uc.pt⟩

mon modeling formalisms are derived, namely, *discrete time systems,* (*Ordinary*), *differential equation systems,* and *discrete event systems.* These formalisms are proved closed under coupling, making a hierarchical representation of systems possible [Zeigler 1976; 1984]. All the formalisms also have a parallel interpretation, making them amenable to parallel implementation.

This work is a generalization of the dynamic structure discrete event system specification (DSDEVS), a formalism to represent discrete event systems that can change their structures dynamically [Barros 1995; Barros 1996a; Barros 1996c] to other types of systems. The DSDE, a parallel version of the original DSDEVS, is presented; the DSDEVS is a rigorous approach to representing dynamic structures in discrete event systems.

This paper is organized into five sections. In Section 2 we summarize the three modeling formalisms developed by Zeigler for representation of basic systems (nonnetwork): discrete time system specification (DTSS); differential equation system specification (DESS); and discrete event system specification (DEVS). These formalisms are used to describe system networks. Section 3 introduces the new concept of *dynamic structure system network.* Section 4 develops the concept of a dynamic system network for DTSS, DESS, and DEVS formalisms. Section 5 presents the conclusions.

2. FORMALISMS FOR BASIC SYSTEMS

To facilitate the understanding of the three dynamic structure formalisms, we first briefly describe the concept of system and summarize three systems specifications. A complete description has been made by Zeigler [1976; 1984]. A system specification provides a shorthand notation to specify classes of systems that have some common properties. A complete system description can be derived from its specification. The procedure for this conversion and the conditions that must be observed can be found in Zeigler [1976]. For convenience, we have extended all the original formalisms to include the model *initial state*.

2.1 System

A *system* is an abstract concept that describes how entities behave over time. It describes output behavior on the basis of inputs and state information. Formally a system is an 8-tuple [Zeigler 1976], $S = (T, X, \Omega, Q, q_0, Y, \delta, \lambda)$, where T is the time base, X is the input value set, Y is the output value set, Ω is the input segment set, Q is the set of states, q_0 is the initial state, δ: $Q \times \Omega \rightarrow Q$ is the transition function, and λ: $Q \rightarrow Y$ is the output function.

The system is subject to the following two constraints [Zeigler 1976]:

(1) Ω is closed under composition;
(2) for every pair of contiguous segments ω and $\omega' \in \Omega$ and for all $q \in Q$, $\delta(q, \omega\omega') = \delta(\delta(q, \omega), \omega')$.

If the time base is the set of real numbers, the system is said to be continuous. If the time base is the set of integers, the system is said to be discrete. A detailed description can be found in Zeigler [1976].

2.2 Discrete Time System Specification

Some systems change their states at regular intervals. Input and output segments are only computed at the end of these intervals. A discrete time system specification provides a short notation to characterize a family of systems with an integer time base. Formally, a *discrete time system specification* is a 6-tuple [Zeigler 1976]: $DTSS = (X, Q, q_0, Y, \delta, \lambda)$, where X is the set of input values, Q is the set of states, q_0 is the initial state, Y is the set of output values, δ: $Q \times X \rightarrow Q$ is the single step transition function, and λ: $Q \rightarrow Y$ is the output function.

At a periodic rate, this model checks its inputs and, based on its state information, produces an output and changes its internal state.

2.3 Differential Equation System Specification

Differential equations are used as a compact notation to describe a large family of dynamic systems. Instead of describing the relation between the input and the state of a system, a differential equation is used to relate the input and the model state derivative. A *differential equation system specification* is a 6-tuple [Zeigler 1976]: $DESS = (X, Q, q_0, Y, f, \lambda)$, where X is the set of input values, Q is the set of internal states, q_0 is the initial state, Y is the set of output values, f: $Q \times X \rightarrow Q$ is the rate of change function, and λ: $Q \rightarrow Y$ is the output function.

The DESS is subjected to the constraints, $X \in \mathbf{R}^l$, $Q \in \mathbf{R}^m$, $Y \in \mathbf{R}^n$ with $l, m, n \in \mathbf{I}_0^+$, and f satisfies the Lipschitz condition.

Let ω: $\langle t_1, t_2 \rangle \rightarrow X$ be a bounded continuous segment, and $q \in Q$ be a state. A segment $\Phi_{q,\omega}$: $\langle t_1, t_2 \rangle \rightarrow Q$ is a solution associated with ω and q if

(1) $\Phi_{q,\omega}(t_1) = q$
(2) $d\Phi_{q,\omega}(t)/dt = f(\Phi_{q,\omega}(t), \omega(t)), t \in \langle t_1, t_2 \rangle$.

The solution for the state trajectory is given for $t \in \mathrm{dom}(\omega) = \langle t_1, t_2 \rangle$, by $\Phi_{q,\omega}(t) = q + \int_{t_1}^{t} f(\Phi_{q,\omega}(t'), \omega(t'))\, dt'$.

In many cases, however, this integral cannot be computed analytically, and simulation is still the best tool.

2.4 Discrete Event System Specification

Some systems change their states a finite number of times in a bounded interval. The time base is the set of nonnegative real numbers, $\mathbf{R}_0^+$. Although the state does not change continuously, the time when these changes occur is a real number. A *discrete event system specification* is a 7-tuple [Zeigler 1976]: $DEVS = (X, S, s_0, Y, \delta, \lambda, \tau)$, where X is the set of input values, S is the set of partial states, s_0 is the initial partial state, Y is the set of output values, and δ: $Q \times (X \cup \{\varnothing\}) \rightarrow S$ is the transition function, where $Q = \{(s, e) | s \in S, 0 \leq e \leq \tau(s)\}$ is the state set, e is the

time elapsed since last transition, $q_0 = (s_0, 0)$ is the initial state, $\varnothing$ is the null value (absence of value), λ: $S \to Y$ is the partial output function, and Λ: $Q \to Y$ is the output function defined by

$$\Lambda(s, e) = \begin{Bmatrix} \lambda(s) & \text{if } e = \tau(s) \\ \varnothing & \text{if } e < \tau(s) \end{Bmatrix}$$

τ: $S \to \mathbf{R}_0^+$ is the time advance function.

If no event arrives at the system, it will stay in partial state s for time $\tau(s)$. When $e = \tau(s)$, the system changes to the state $(\delta(s, \tau(s), \varnothing), 0)$. If an external event, $x \in X$, arrives when the system is in the state (s, e) it will change to the state $(\delta(s, e, x), 0)$. If an external event arrives when $e = \tau(s)$, the system changes to the state $(\delta(s, \tau(s), x), 0)$. A DEVS can only produce an output when $e = \tau(s)$; in all other cases there is no output, i.e., $\Lambda(s, e) = \varnothing$, if $e < \tau(s)$. A DEVS specifies a system only if it is *legitimate* [Zeigler 1976].

3. DYNAMIC STRUCTURE SYSTEM NETWORK

A heuristic rule to handle complex systems is to decompose them into less complex ones. Such a combination of systems is called a *system network.* Networks described by conventional formalisms have a static structure [Mesarovic and Takahara 1975; Wymore 1977; Zeigler 1976]. The real world can, in some cases, be better represented by system networks that undergo structural changes. We introduce the *dynamic structure system network,* a new concept that can change its structure dynamically. To simplify the presentation, we describe networks without inputs and outputs only.

A *dynamic structure system network* is a tuple, $DSSN = (\chi, S_\chi)$, where χ is the *network executive* name, and S_χ is the system describing the executive χ.

The dynamic structure system network is defined with a special component, the *network executive* χ. S_χ, the model of the executive, is a modified system defined by the 10-tuple, $S_\chi = (T, X_\chi, \Omega_\chi, Q_\chi, q_{0,\chi}, Y_\chi, \gamma, \Sigma^*, \delta_\chi, \lambda_\chi)$, where γ: $Q_\chi \to \Sigma^*$ is the structure function, and Σ^* is the set of network structures.

The meaning of the other variables was previously described in Section 2.1.

A structure $\Sigma \in \Sigma^*$ at a state $q_\chi \in Q_\chi$ is given by $\Sigma = \gamma(q_\chi) = (D, \{S_i\}, \{I_i\}, \{Z_i\})$, where D is the set of component (system) names. For all $i \in D$, S_i is a system component i, I_i is the set of influencers of component i, and Z_i is the i-input function.

The state variables are subjected to the following constraints: $\chi \notin D$, for all $i \in D$, $S_i = (T, X_i, \Omega_i, Q_i, q_{0,i}, Y_i, \delta_i, \lambda_i)$ is a system, and Z_i: $\times_{j \in I_i} Y_j \to X_i$.

Because the network coupling information is located in the state of the executive, transition functions can change this state and, in consequence, change the structure of the network.

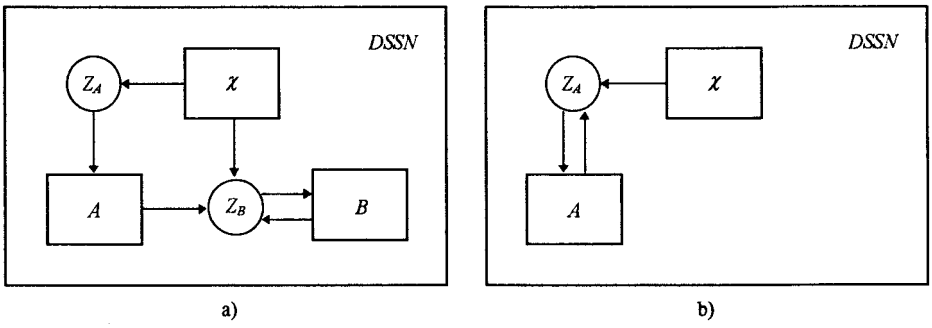

Fig. 1. Dynamic structure network.

Definition 1. The 4-tuple $(D, \{S_i\}, \{I_i\}, \{Z_i\}) = \gamma(q_\chi)$, $q_\chi \in Q_\chi$, is referred to as the *network structure.*

Definition 2. *Any* change in the network structure $(D, \{S_i\}, \{I_i\}, \{Z_i\})$, is defined as a *change of structure.*

Changes in structure are defined in a broad sense, including changes in component interconnection, changes in system definition, and the addition or deletion of system components.

Example. Figure 1(a) describes a dynamic structure network. From the diagram, we can infer the structural properties of the network, namely, its composition and the interaction among systems. This network is defined by $DSSN = (\chi, S_\chi = (T, X_\chi, \Omega_\chi, Q_\chi, q_{0,\chi}, Y_\chi, \gamma, \Sigma^*, \delta_\chi, \lambda_\chi))$.

If Figure 1(a) represents the initial structure of the network, then $\gamma(q_{0,\chi}) = (D, \{S_i\}, \{I_i\}, \{Z_i\})$, where $D = \{A, B\}$, $S_A = (T, X_A, \Omega_A, Q_A, q_{0,A}, Y_A, \delta_A, \lambda_A)$, $S_B = (T, X_B, \Omega_B, Q_B, q_{0,B}, Y_B, \delta_B, \lambda_B)$, $I_\chi = \{\}$, $I_A = \{\chi\}$, $I_B = \{\chi, A, B\}$, Z_A: $Y_\chi \to X_A$, and Z_B: $Y_\chi \times Y_A \times Y_B \to X_B$.

The executive state after receiving the segment $\omega_\chi \in \Omega_\chi$ is given by $q'_\chi = \delta_\chi(q_\chi, \omega_\chi)$, and the new network structure is given by $\gamma(q'_\chi) = (D', \{S'_i\}, \{I'_i\}, \{Z'_i\})$, where $D' = \{A\}$, $S'_A = (T, X_A, \Omega_A, Q_A, q_{0,A}, Y_A, \delta'_A, \lambda_A)$, $I'_\chi = \{\}$, $I'_A = \{\chi, A\}$, and Z'_A: $Y_\chi \times Y_A \to X_A$.

The block diagram corresponding to this new structure is given in Figure 1(b). Changes in structure include the deletion of system component *B,* the change of the transition function of component *A* from δ_A to δ'_A, the change of the influencers of *A,* and change of the input function of component *A.*

When the executive modifies its state, the network structure can also be modified. Because the network structure is stored in the executive state, any kind of structural change can be achieved. Namely, changes in

—components, represented by D;

—system representation of components, represented by $\{S_i\}$; and

—connections among components, represented by $\{Z_i\}$ and $\{I_i\}$.

The structure can change not only by the addition or deletion of components, but also by changing the model definition and connections as well.

Table I. Modeling Formalisms for Dynamic Structure Systems

	FORMALISMS		
LEVELS OF SYSTEM SPECIFICATION	Discrete Time (DSDT)	Differential Equation (DSDQ)	Discrete Event (DSDE)
Dynamic Structure System Network	DSDTN	DSDQN	DSDEN
System	DTSS	DESS	DEVS

The set of added and removed components in each transition depends on the application domain. This information is stored in the state variables and within the transition functions. Models can be kept in the executive state variables, but as long as they do not belong to set $D = \text{proj}_D(\gamma(s_\chi))$, they are not components of the network.

4. MODELING FORMALISMS FOR DYNAMIC STRUCTURE NETWORKS

We have seen that system specifications provide a compact and simple notation to describe basic systems. We now develop system specifications to describe a network of systems. Table I represents both the basic and the network system specifications. The new formalisms use the same representation for basic systems, but use the concept of *dynamic structure network* to describe special subclasses of system networks.

We have created three formalisms, DSDT, DSDQ, and DSDE, to represent networks of the previously described models—respectively, DTSS, DESS, and DEVS. The DSDE formalism presented here is a modified version of an earlier version [Barros 1996]. This formalism defines its behavior in a parallel way, making such networks more amenable to an implementation in a parallel machine.

To build complex models it is necessary to use formalisms that support hierarchical and modular models. Models can thus be decomposed into other models in a recursive way. This kind of modular construction is only possible if formalisms are *closed under coupling*.

Definition 3 [Zeigler 1984]. A formalism is *closed under coupling* if any network obtained by coupling components specified by the formalism can also be specified by the formalism.

To show the correctness and the usefulness of the new modeling formalisms, we demonstrate that they are closed under coupling.

4.1 Dynamic Structure Discrete Time System Specification

The dynamic structure discrete time system specification is a formalism to describe basic or network discrete time systems. The DSDT basic model is

the DTSS model described in Section 2.1. The network of simple DTSS models is referred to as a *dynamic structure discrete time system network.* Formally, a dynamic structure discrete time system network is a tuple, $DSDTN = (\chi, M_\chi)$, where χ is the name of the *dynamic structure network executive,* and M_χ is the model of the executive χ.

The model of the executive is a modified DTSS defined by the 8-tuple, $DTSS_\chi = (X_\chi, Q_\chi, q_{0,\chi}, Y_\chi, \gamma, \Sigma^*, \delta_\chi, \lambda_\chi)$.

The network structure Σ, at a state $q_\chi \in Q_\chi$, is given by $\Sigma = \gamma(q_\chi) = (D, \{M_i\}, \{I_i\}, \{Z_i\})$, where for all $i \in D$, $M_i = (X_i, Q_i, q_{0,i}, Y_i, \delta_i, \lambda_i)$ is a DTSS.

THEOREM 1. *The DSDT formalism is closed under coupling; that is, the* $DSDTN = (\chi, M_\chi)$ *is equivalent to the* $DTSS = (Q, q_0, \delta)$.

PROOF. We describe the *DTSS* in terms of the elements in the *DSDTN.* The state set Q is given by $Q = \cup_{q_\chi \in Q_\chi} (\times_{i \in D_\chi} Q_i)$, where D_χ, the set of components associated with a state q_χ, is given by $D_\chi = \text{proj}_D(\gamma(q_\chi)) \cup \{\chi\}$, and the model of component $i \in D_\chi$, is given by $M_i = (X_i, Q_i, q_{0,i}, Y_i, \delta_i, \lambda_i)$.

Note that the state set Q is the union of all possible states that might exist in a network; that is, this set contains all the possible states of the network for all possible combinations of components. So the transition function δ maps values from Q to Q.

The initial state, $q_0 \in Q$, is given by $q_0 = \times_{i \in D_{0,\chi}} q_{0,i}$, where $D_{0,\chi}$, the initial set of network components, is given by $D_{0,\chi} = \text{proj}_D(\gamma(q_{0,\chi})) \cup \{\chi\}$.

To define the transition function δ: $Q \to Q$, let $q_\chi \in Q_\chi$ be the current executive state and Σ the current network structure, $\Sigma = \gamma(q_\chi) = (D, \{M_i\}, \{I_i\}, \{Z_i\})$.

The new executive state $q'_\chi \in Q_\chi$ is given by $q'_\chi = \delta_\chi(q_\chi, x_\chi)$, where the input x_χ of the executive is given by $x_\chi = Z_\chi(\times_{d \in I_\chi} \lambda_d(q_d))$.

After the executive transition, the new network structure Σ' is given by $\Sigma' = \gamma(q'_\chi) = (D', \{M'_i\}, \{I'_i\}, \{Z'_i\})$.

The transition function is thus defined by $\delta(\times_{i \in D \cup \{\chi\}} q_i) = \times_{d \in D' \cup \{\chi\}} q_d$, where the new state $q'_j \in Q'_j$ of each component $j \in D'$ (the new set of network components) is given by

$$q'_j = \begin{cases} \delta_j(q_j, x_j) & \text{if } j \in K \quad (1.1) \\ q_{0,j} & \text{if } j \in A \quad (1.2) \end{cases}$$

and the input x_j of component j is given by $x_j = Z_j(\times_{d \in I_j} \lambda_d(q_d))$.

A is the set of the new (added) components, $A = D' - D$, and K is the set of the kept components, $K = D \cap D'$.

Line 1.1 of the definition computes the next state of systems that are not removed from the network. The next state is obtained using the current structure of the network, i.e., $(D, \{M_i\}, \{I_i\}, \{Z_i\})$. The network structure $(D', \{M'_i\}, \{I'_i\}, \{Z'_i\})$, will only be used in the next step. Line 1.2 of the definition states that the new added components have an initial state given in their definition.

We emphasize that there is no ambiguity in the formalism. All the outputs are taken simultaneously, and the current network structure (D, $\{M_i\}$, $\{I_i\}$, $\{Z_i\}$) is used to compute the next state. The new structure (D', $\{M_i'\}$, $\{I_i'\}$, $\{Z_i'\}$) is only used in the next transition. If the output values were not taken at the same time, their processing order will lead to different results. If, for example, the executive inputs were processed first, the network structure could be changed and the other input values would act on a different network.

4.2 Dynamic Structure Differential Equation System Specification

The *dynamic structure (ordinary) differential equation system specification* is a formalism to specify basic or network differential equation systems. The DSDQ basic model is the DESS model described in Section 2.2. The network of simple DESS models is referred to here as the *dynamic structure differential equation system network.* A dynamic structure differential equation network is a tuple, $DSDQN = (\chi, M_\chi)$.

The model of the executive is a modified DESS and is defined by $DESS_\chi = (X_\chi, Q_\chi, q_{0,\chi}, Y_\chi, \gamma, \Sigma^*, f_\chi, \lambda_\chi)$, subject to the constraint that γ is a continuous function.

The network structure, at a state $q_\chi \in Q_\chi$, is given by the tuple, $\gamma(q_\chi) = (D, \{M_i\}, \{I_i\}, \{Z_i\})$, where for all $i \in D$, $M_i = (X_i, Q_i, q_{0,i}, Y_i, f_i, \lambda_i)$ is a DESS.

Due to its discrete nature, the set D of components cannot be changed continuously, so the set D must remain unchanged. Also, the equivalent basic model of a network model has a state given by the cross product of states of the components. So adding and removing components will add or remove state variables. As the creation or the destruction of a state variable is a discontinuity, these types of systems cannot be described by differential equations satisfying the Lipschitz condition. However, adding and removing components can be represented by hybrid (continuous-discrete) models.

For the overall model to be described by a differential equation satisfying the Lipschitz condition, the structure function γ must be a continuous function.

Although components must remain the same, their definition, represented by $\{M_i\}$, can be changed continuously. These changes must follow some constraints. As described in Section 2.3, a DESS M is given by $M = (X, Q, q_0, Y, f, \lambda)$, and the sets X, Q and Y are subjected to the constraints, $X \in \mathbf{R}^l$, $Q \in \mathbf{R}^m$, $Y \in \mathbf{R}^n$ with $l, m, n \in \mathbf{I}_0^+$.

The values of l, m, and n cannot, obviously, be changed continuously, and consequently the sets X, Q, and Y cannot be changed. However, the values of f and λ can be changed continuously, as they represent functions of real variables. The change of the initial state q_0 can be achieved, but is of little interest because it is only used once at the time each component is placed in the network.

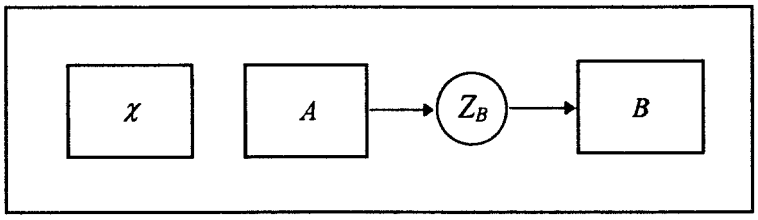

Fig. 2. Block diagram of the dynamic structure differential equation network.

Changes of structure in continuous systems also include the change of the input function Z_i, of component i. The change of the influencers of a component i, I_i, cannot be achieved due to the discrete nature of this set.

THEOREM 2. *The DSDQ formalism is closed under coupling; that is, the* $DSDQN = (\chi, M_\chi)$ *is equivalent to the DESS* $= (Q, q_0, f)$.

PROOF. We describe the *DESS* in terms of the elements in the *DSDQN*.

The state set Q is given by $Q = \times_{i \in D_\chi} Q_i$, where D_χ, the set of components associated with any state $q_\chi \in Q_\chi$, is given by $D_\chi = \text{proj}_D(\gamma(q_\chi)) \cup \{\chi\}$.

The initial state, $q_0 \in Q$, is given by $q_0 = \times_{i \in D_\chi} q_{0,i}$.

To define the rate of change function f: $Q \to Q$, let the current network structure by given by $\Sigma = \gamma(q_\chi) = (D, \{M_i\}, \{I_i\}, \{Z_i\})$ and the new network structure by $\Sigma' = \gamma(q'_\chi) = (D, \{M'_i\}, \{I'_i\}, \{Z'_i\})$.

The rate of change function is defined by $f(\times_{i \in D_\chi} q_i) = \times_{i \in D_\chi} q'_i$, where the new state $q'_i \in Q_i$, *of each component* $i \in D_\chi$ is given by $q'_i = \Phi_{q_i,\omega_i}(t_1 + \epsilon)$, $\lim \epsilon \to 0$, and for each $i \in D_\chi$, ω_i: $\langle t_1, t_1 + \epsilon \rangle \to X_i$, is a bounded continuous segment and Φ_{q_i,ω_i}: $\langle t_1, t_1 + \epsilon \rangle \to Q_i$ a solution associated with ω_i and q_i, such that $\Phi_{q_i,\omega_i}(t_1) = q_i$ and $d\Phi_{q_i,\omega_i}(t)/dt = f_i(\Phi_{q_i,\omega_i}(t), \omega_i(t))$, for $t \in \langle t_1, t_1 + \epsilon \rangle$ and the input segment, ω_i is given, for $t \in \text{dom}(\omega_i) = \langle t_1, t_1 + \epsilon \rangle$, by $\omega_i(t) = Z_i(\times_{d \in I_i} \lambda_d(\Phi_{q_d,\omega_d}(t)))$.

The next state is computed using the current structure of the network $(D, \{M_i\}, \{I_i\}, \{Z_i\})$. The new network structure $(D, \{M'_i\}, \{I'_i\}, \{Z'_i\})$, will only be used at time $t_1 + \epsilon$.

Example. Consider a dynamic structure system network that changes its structure continuously by adapting the input, output, and rate of change function of one of its components. The network in Figure 2 is a *DSDQN,* and its description is given by $DSDQN = (\chi, M_\chi)$.

The model of the executive is a DESS defined by $DESS_\chi = (Q_\chi, q_{0,\chi}, \gamma, \Sigma^*, f_\chi)$, where $Q_\chi = \mathbf{R}^3$ and f_χ: $Q_\chi \to Q_\chi$.

We consider, in this example, a rate of change function given by $f_\chi((a, b, c)) = (-a^2, -b, -c^3)$, where a state $q_\chi \in Q_\chi$ is given by $q_\chi = (a, b, c)$.

The network structure at state $q_\chi \in Q_\chi$ is given by $\Sigma = \gamma(q_\chi) = (D, \{M_i\}, \{I_i\}, \{Z_i\})$, where $D = \{A, B\}$, $M_A = (Q_A, q_{0,A}, Y_A, f_A, \lambda_A)$, and $M_B = (X_B, Q_B, q_{0,B}, f_B)$, with $Y_A = X_B = Q_A = Q_B = \mathbf{R}$, $\lambda_A = a\ \lambda'_A$, and $f_A = b\ f'_A$, where f'_A is a function satisfying the Lipschitz condition, $\{I_\chi\} = \{I_A\} = \{\}$ and $I_B = \{A\}$.

The input function of model B is expressed by Z_B: $Y_A \rightarrow X_B$, and is defined by $Z_B(y) = c\ y$.

The values of *a, b,* and *c* are controlled by the differential equation f_χ, and thus they change continuously. We emphasize that for each value of the triple (*a, b, c*), and by Definition 2, there is a different network. We note that changes in model input function, for example, cannot be made in static structure models.

In general, all problems can be reduced to basic models, and structure variation does not increase the power of representation of any formalism. However, structure in general, and dynamic structures in particular, can in many cases lead to models that are easier to understand and implement.

4.3 Parallel Dynamic Structure Discrete Event System Specification

The problem of representing discrete event systems that undergo structural changes has been the subject of many research papers Thomas 1994; Uhrmacher and Arnold 1994; Vasconcelos 1993; Zeigler and Reynolds 1985; Zeigler and Praehofer 1989; Zeigler et al. 1991]. However, these approaches do not achieve a general and formal framework for this problem. A rigorous approach is the DSDEVS formalism [Barros 1995; 1996a]. A comparison of the several methods used to represent dynamic structure discrete event systems can be found in Barros [1997b]. The *parallel dynamic structure discrete event system specification* is a generalization of the original DSDEVS formalism, and allows the specification of dynamic structure networks of discrete event systems. The DSDE basic model is the DEVS described in Section 2.3. The network of simple DEVS models is referred to as the *parallel dynamic structure discrete event system network.* Formally, a parallel dynamic structure discrete event system network is a tuple, $DSDEN = (\chi, M_\chi)$, where χ is the name of the *dynamic structure network executive* and M_χ is the model of the executive χ.

The model of the executive is a modified DEVS, defined by the 9-tuple, $M_\chi = (X_\chi, S_\chi, s_{0,\chi}, Y_\chi, \gamma, \Sigma^*, \delta_\chi, \lambda_\chi, \tau_\chi)$.

The network structure $\Sigma \in \Sigma^*$, at a state $s_\chi \in S_\chi$, is given by the tuple $\Sigma = \gamma(s_\chi) = (D, \{M_i\}, \{I_i\}, \{Z_i\})$, where for all $i \in D$, $M_i = (X_i, S_i, s_{0,i}, Y_i, \delta_i, \lambda_i, \tau_i)$ is a DEVS, and $Z_i(\times_{|I_i|} \varnothing) = \varnothing$.

The last constraint states that if all the influences of a component have a null output value, then the component will receive a null input; that is, the input function does not create a value from an absence of values. This behavior is universally accepted for discrete event systems.

THEOREM 3. The DSDE formalism is closed under coupling; that is, the $DSDEN = (\chi, M_\chi)$ is equivalent to a basic model $DEVS = (S, s_0, \delta, \tau)$.

PROOF. We describe the *DEVS* in terms of the elements in the *DSDEN.*

The partial state set S is given by $S = \cup_{s_\chi \in S_\chi} (\times_{i \in D_\chi} Q_i)$, where D_χ, the set of components associated with the current partial state s_χ, is given by $D_\chi = \text{proj}_D(\gamma(s_\chi)) \cup \{\chi\}$, and the state set of a component i is defined by $Q_i = \{(s_i, e_i) | s_i \in S_i, 0 \leq e_i \leq \tau_i(s_i)\}$.

We define σ_i (the time component i must still remain in the current state) by $\sigma_i = \tau_i(s_i) - e_i$.

The initial partial state is given by $s_0 = \times_{i \in D_{0,\chi}} q_{0,i}$, where $D_{0,\chi}$, the initial set of network components, is given by $D_{0,\chi} = \text{proj}_D(\gamma(s_{0,\chi})) \cup \{\chi\}$.

The time advance function is given by τ: $S \rightarrow \mathbf{R}_0^+$, and is defined by $\tau(s) = \min \{\sigma_i | i \in D_\chi\}$.

The set of states Q is given by $Q = \{(s, e) | s \in S, 0 \leq e \leq \tau(s)\}$.

To define the transition function δ: $Q \rightarrow S$, let the current network structure $\Sigma \in \Sigma^*$, at the current state $s_\chi \in S_\chi$, be given by $\Sigma = \gamma(s_\chi) = (D, \{M_i\}, \{I_i\}, \{Z_i\})$, and let Σ' be the new network structure associated with the new executive state $s'_\chi \in S_\chi$, $\Sigma' = \gamma(s'_\chi) = (D', \{M'_i\}, \{I'_i\}, \{Z'_i\})$.

Thus the transition function is defined by $\delta(\times_{i \in D_\chi} q_i, e) = \times_{j \in D'_\chi} q_j$, where the new set of network components (including the executive) is given by $D'_\chi = \text{proj}_D(\gamma(s'_\chi)) \cup \{\chi\}$, and the new state $q'_j \in Q'_j$ of each component $j \in D'_\chi$, is given by

$$q'_j = \begin{cases} (\delta_j(s_j, e_j + \tau(s), x_j), 0) & \text{if } j \in D_\chi \cap D'_\chi \wedge (x_j \neq \varnothing \vee \sigma_j = \tau(s)) \quad (3.1) \\ (s_j, e_j + \tau(s)) & \text{if } j \in D_\chi \cap D'_\chi \wedge x_j = \varnothing \wedge \sigma_j > \tau(s) \quad (3.2) \\ q_{0,j} & \text{if } j \in D'_\chi - D_\chi \quad (3.3) \end{cases}$$

with $x_j = Z_j\ (\times_{d \in I_j} \Lambda_d\ (q_d))$.

Line 3.1 of the definition computes the next state of the models that either receive an external input or are scheduled to change; line 3.2 computes the next state of the remaining components. These models only update their elapsed time.

All inputs to a model are considered *simultaneously,* making this interpretation more amenable to a *parallel* implementation. In an earlier version, the DSDEVS formalism imposed the condition that inputs from different models could only be considered one at a time, *serialized* by a *select function.*

Line 3.3 of the definition states that the new added components start in their own initial state.

Example. Consider the simple network in Figure 3. This model represents a flow-shop, in its initial structure, with one workstation ($W1$) and one product generator (G). At scheduled times, the generator shifts between the current product and a new product. This layout alteration is performed by the flow-shop executive upon generator request. The two types of products differ in one operation made by workstation $W2$. When a product requires two operations, the visit to $W2$ follows the process in $W1$. The flow-shop network is defined by $FS = (\chi, M_\chi)$, where $M_\chi = (X_\chi, S_\chi, s_{0,\chi}, \gamma, \Sigma^*, \delta_\chi, \tau_\chi)$, $X_\chi = \{\textbf{change}\}$, and $\tau_\chi(s_\chi) = \infty$.

The executive does not produce any output, it only reacts to external messages to change the network structure. As a consequence, an executive output function can be omitted. The initial structure is given by $\Sigma =$

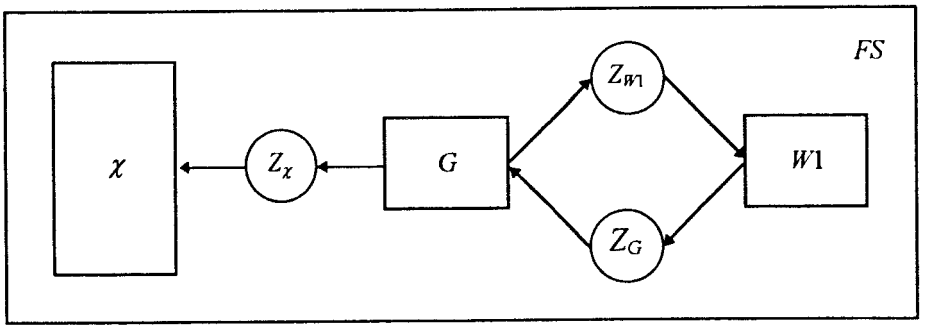

Fig. 3. Single workstation network.

$\gamma(s_{0,\chi}) = (D, \{M_i\}, \{I_i\}, \{Z_i\})$, where $D = \{G, W1\}$, M_G and M_{W1} are DEVS models, $I_G = \{W1\}$, $I_{W1} = \{G\}$, $I_\chi = \{G\}$, and Z_G: $Y_{W1} \rightarrow X_G$, Z_{W1}: $Y_G \rightarrow X_{W1}$, Z_χ: $Y_G \rightarrow X_\chi$.

This network is depicted in Figure 3. We consider this the initial flow-shop configuration. The first change command will transform this model into a two-workstation flow-shop. At scheduled times the generator G changes the product type by issuing a message to the executive to add or remove one workstation to the shop floor. The executive will then commute to another flow-shop layout. Products will differ because they do not undergo the same set of operations. The transition function is defined by $\delta(s_{0,\chi}, e, \textbf{change}) = s'$.

The new executive state represents a flow-shop with two workstations; that is, $\Sigma' = \gamma(s') = (D', \{M_i'\}, \{I_i'\}, \{Z_i'\})$, with $D' = \{G, W1, W2\}$, M_G, M_{W1} and M_{W2} are DEVS models, $I_G = \{W2\}$, $I_{W1} = \{G\}$, $I_{W2} = \{W1\}$, $I_\chi = \{G\}$, and Z_G: $Y_{W2} \rightarrow X_G$, Z_χ: $Y_G \rightarrow X_\chi$, Z_{W1}: $Y_G \rightarrow X_{W1}$, Z_{W2}: $Y_{W1} \rightarrow X_{W2}$.

The executive state is changed by the addition of one workstation, W2. The influencers' set and the output functions were also modified. The new flow-shop is represented in Figure 4. The next executive state is given by $\delta(s', e, \textbf{change}) = s''$.

The new executive state represents a flow-shop at the initial configuration; that is $\Sigma = \gamma(s'') = (D, \{M_i\}, \{I_i\}, \{Z_i\})$.

4.4 Parallel Formalisms

The DSDEVS, an earlier version of the DSDE formalism, is described in Barros [1995] and Barros [1996a], and was implemented in the DELTA modeling and simulation environment [Barros 1996b]. However, the DSDEVS does not capture the parallel nature of discrete event systems. The formalism uses a select function to choose and to change only one model at each time.

In the DSDE formalism, the select function is removed by considering for each model its influencers instead of its influencees. This allows all the inputs to every model to be considered simultaneously, i.e., in parallel.

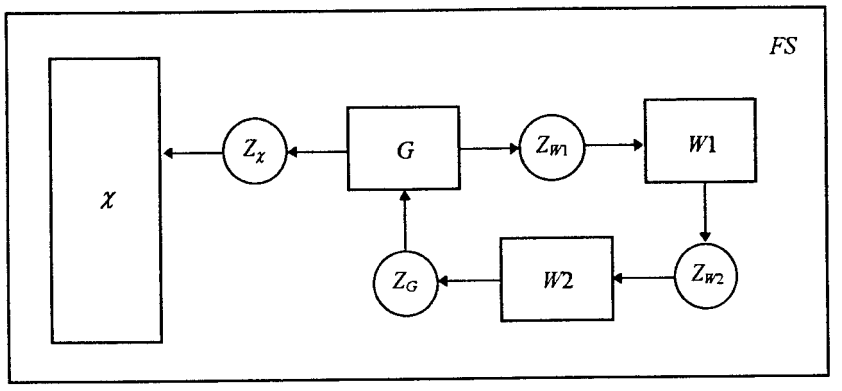

Fig. 4. A two-workstation flow-shop.

The P(arallel)-DEVS, a formalism to capture the parallelism in static structure discrete event systems, is described in Chow and Zeigler [1994]. This formalism also removes tie breaks. However, it still keeps the influences of each model, and does not allow the existence of model self-loops. A more detailed comparison of the DSDE and P-DEVS formalisms is not in the scope of this paper, and can be found in Barros [1997a].

Parallelism can be seen at two levels: (1) implementation and (2) formalism. At the former level, all modeling formalisms allow some type of parallelization. We refer to parallelism at the formalism level only. Although the original differential equation and discrete time networks are inherently parallel, discrete event networks were designed to be sequential [Zeigler 1976]. To our knowledge, the P-DEVS and the DSDE are the only discrete event formalisms that work in parallel; i.e., that do not have a tie break function to select just one event when several occur at the same time.

All the network formalisms rely on the definition of the influencers of a model. However, the formalisms for static structure models have two different types of definitions [Zeigler 1976]. Differential equations and discrete time networks are based on the set of influencers; but discrete event networks are defined with the set of influences. Thus the first two networks are parallel in nature, while discrete event networks must be interpreted in a sequential, i.e., nonparallel, way.

4.5 Modeling Heterogeneous Systems

Current modeling and simulation methodology does not provide a common definition of networks for the various types of system specifications [Praehofer 1991]. This situation makes it difficult to integrate heterogeneous models. Here we outline the benefits of the methodology just developed for mixing heterogeneous systems.

Although there are many methods to integrate differential equations, they are all variations of sampling methods. In these methods, components are sampled at some rate (usually fixed), and a numerical integration method is used to compute the descriptive variables. From the computa-

tional perspective, these methods have been implemented by discrete time machines.

A discrete time system can be viewed as a special case of a discrete event system with a constant time advance function. However, combining these two systems is not easy, since in the previous simulation methodology they had a different network definition. In the methodology presented here, integration of discrete time and discrete event systems is easier to accomplish because all types of networks share the same definition.

5. CONCLUSIONS

We have described three new modeling formalisms for the specification of dynamic structure system networks: (1) *dynamic structure discrete time system specification,* (2) *parallel dynamic structure discrete event system specification,* and (3) *dynamic structure differential equation system specification.* We proved that these formalisms are closed under coupling, and can thus be used to build models in a hierarchical and modular way. Their parallel nature makes them amenable to implementation on parallel computers. Research is currently being developed to make a computational implementation of the formalisms.

REFERENCES

BARROS, F. J. 1995. Dynamic structure discrete event system specification: A new formalism for dynamic structure modeling and simulation. In *Proceedings of the 1995 Winter Simulation Conference* (Arlington, VA), 781–785.

BARROS, F. J. 1996a. Dynamic structure discrete event system specification: Formalism, abstract simulators and applications. *Trans. Soc. Comput. Simul. 13,* 1, 35–46.

BARROS, F. J. 1996b. Structural inheritance in the DELTA environment. In *Proceedings of the Sixth Annual Conference on AI, Simulation and Planning in High Autonomy Systems* (La Jolla, CA, 141–147.

BARROS, F. J. 1996c. Modeling and simulation of dynamic structure discrete event systems: A general systems theory approach. Ph.D. dissertation, Dept. of Informatics Engineering, Univ. of Coimbra.

BARROS, F. J. 1997a. Comparison of two parallel discrete event formalisms, (In preparation).

BARROS, F. J. 1997b. Dynamic structure discrete event systems: A comparison of methodologies and environments. In *Proceedings of SPIE 11th Annual International Symposium on Aerospace/Defense Sensing, Simulation and Controls: Enabling Technology of Simulation Science* (Orlando, FL), Vol. 3083, 268–277.

CHOW, A. C. AND ZEIGLER, B. P. 1994. Abstract simulator for the parallel DEVS formalism. In *Proceedings of the Fifth Annual Conference on AI, Simulation and Planning in High Autonomy Systems* (Gainesville, FL), 157–163.

MESAROVIC, M. D. AND TAKAHARA, Y. 1975. *General Systems Theory: A Mathematical Foundation.* Academic Press, New York, NY.

PRAEHOFER, H. 1991. *System theoretic foundations for combined discrete-continuous system simulation.* Ph.D. dissertation, Dept. of Systems Theory and Information Engineering, Univ. of Linz.

THOMAS, C. 1994. Interface-oriented classification of DEVS models. In *Proceedings of the Fifth Annual Conference on AI, Simulation and Planning in High Autonomy Systems* (Gainesville, FL), 208–213.

UHRMACHER, A. M. AND ARNOLD, R. 1994. Distributing and maintaining knowledge: Agents in variable structure environment. In *Proceedings of the Fifth Annual Conference on AI, Simulation and Planning in High Autonomy Systems* (Gainesville, FL), 178–184.

VASCONCELOS, M. J. 1993. *Modeling spatial dynamic ecological processes with DEVS-scheme and geographic information systems.* Ph.D. thesis, School of Renewable and Natural Resources, Univ. of Arizona.

WYMORE, A. W. 1977. *A Mathematical Theory of Systems Engineering: The Elements.* Krieger, New York, NY.

ZEIGLER, B. P. 1976. *Theory of Modelling and Simulation,* Wiley, New York, NY.

ZEIGLER, B. P. 1984. *Multifaceted Modelling and Discrete Event Simulation.* Academic Press, London, UK.

ZEIGLER, B. P. AND REYNOLDS, R. G. 1985. Towards a theory of dynamic computer architectures. In *Proceedings of the 5th International Conference on Distributed Computing Systems.* Computer Society Press, 468–475.

ZEIGLER, B. P. AND PRAEHOFER, H. 1989. Systems theory challenges in the simulation of variable structure and intelligent systems. In *CAST-Computer-Aided Systems Theory,* F. Pichler and Moreno-Diaz, Eds. Springer Verlag, Berlin, 41–51.

ZEIGLER, B. P., KIM, T. G., AND LEE, C. 1991. Variable structure modelling methodology: A dynamic computer architecture example. In *Trans. Soc. Comput. Simul. 7,* 4, 291–319.

Received March 1996; revised March 1997; accepted September 1997