International Journal of General Systems
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713642931

EXPERIMENTAL FRAME SPECIFICATION METHODOLOGY FOR HIERARCHICAL SIMULATION MODELING

Jerzy W. Rozenblit[a]
[a] Department of Electrical and Computer Engineering, The University of Arizona, Tucson, Arizona, USA

To cite this Article Rozenblit, Jerzy W.(1991) 'EXPERIMENTAL FRAME SPECIFICATION METHODOLOGY FOR HIERARCHICAL SIMULATION MODELING', International Journal of General Systems, 19: 3, 317 — 336
To link to this Article: DOI: 10.1080/03081079108935180
URL: http://dx.doi.org/10.1080/03081079108935180

Int. J. General Systems, Vol. 19, pp. 317–336
Reprints available directly from the publisher
Photocopying permitted by license only

EXPERIMENTAL FRAME SPECIFICATION METHODOLOGY FOR HIERARCHICAL SIMULATION MODELING

JERZY W. ROZENBLIT

Department of Electrical and Computer Engineering, The University of Arizona, Tucson, Arizona 85721 USA

(*Received September 1990; in final form 8 February 1991*)

A methodology is presented for defining conditions under which simulation models can be observed and experimented with. Such conditions are formalized as *experimental frames*. A method for deriving experimental frame specifications for simulation models is given based on a universal specification called *generic experimental frame*. The methodology defines two schemes for carrying out simulation experiments with hierarchical, modular models: 1) the centralized architecture is based on a global experimental frame which specifies conditions for the entire model; 2) the distributed architecture facilitates attachments of frame components to model simulators at different levels of the model hierarchy. An example of a simple manufacturing process illustrates the conceptual framework. Implications of the proposed framework for design of high autonomy systems are discussed as well.

INDEX TERMS: Simulation modeling, experimental frame, hierarchical, modular systems, distributed simulation

1. INTRODUCTION

Simulation modeling is a professional, intellectual, and academic discipline whose primary concerns are the construction of real world systems' models, computer simulation of the models, and analysis of simulation results. While the discipline itself has been the subject of extensive theoretical and experimental studies,[1–3] its ultimate benefit is to improve decision making in engineering, business and other environments. Thus, simulation modeling should be an inherent component in computer-aided decision systems in management, control, and design.

The tools and activities prescribed by simulation methodologies are intended to assist designers in evaluating alternative designs.[2] Consider, for instance, a robot task and motion planner which attempts to prescribe a time-optimal, collision free trajectory. Among other factors, the trajectory depends on parameters of the robot's control program instructions (e.g., speed of motion, acceleration, etc.). It is desirable that the computed trajectory be verified by simulation prior to its execution in a real system. The consequences of erroneous robot's actions in, for example, a factory setting are easy to imagine; they may range from delays in processing a detail to a serious damage to either the robot or the equipment. If the trajectory is simulated and the results are not satisfactory, program parameters can be changed and a new trajectory can be studied in a subsequent simulation run. The need for simulation-based robot action planning is being increasingly recognized.[4,5]

The choice of performance measures under which a system is evaluated reflects the objectives of the designer. The nature of objectives guides the development of

both models and simulation experiments.[2,3,6] Any real system could be subjected to a multiplicity of objectives in a management, control, and design context. Envision an automated laboratory in which robots perform a variety of manufacturing tasks. Several aspects of the laboratory's operation need to be evaluated before it is set up. For example, to calculate robots' motion trajectories, we require a model of their workscene as well as the experimental conditions under which the robots perform their actions; computing the utilization of equipment requires models of the equipment and experiments measuring the ratio of the time the equipment is in service versus the time it is idle. Thus objectives of simulation orient both the model and experiment construction.

In this article, we focus on the experimentation aspect of simulation. The key concept of this aspect is *experimental frame*, i.e., the specification of circumstances under which a model (or a real system) is observed and experimented with.[3] An experimental frame reflects modeling objectives since: a) it subjects a model to input stimuli (which represent potential interventions into the model's operation); b) it observes the model's reactions to the input stimuli and collects the data about such reactions (output data); and 3) it controls the experimentation by placing relevant constraints on values of the designated model state variables and by monitoring these constraints.

Generation of meaningful experimental conditions is a complex task requiring that the modeller understand the nature of the objectives and their interactions. An experimental frame, similarly to a model, may reflect a single objective or a complex set of simulation goals. As the complexity of simulation models of large scale systems grows, so does the complexity of related experiments. Whereas a significant progress has been made in improving the model construction process, relatively few authors have focused on the experimental frame definition.[1,3,7-9] This article presents a methodology for developing simulation experiments. Distinct emphasis is placed on the specification of experimental frames in simulation of hierarchical, modular systems.[3,6,10]

The article is organized as follows: first, background information is provided that introduces formally the experimental frame definition. Then, we propose a novel scheme for distributing experiments in a hierarchical simulation environment. The resulting experimental frame architecture is discussed and illustrated with an example of a manufacturing process. Implications of the proposed framework to design of high autonomy systems are drawn at the conclusion.

2. EXPERIMENTAL FRAME DEFINITION

Zeigler[3] has laid down the groundwork for a methodology in which the statement of modeling objectives is operationalized in the definition of experimental frames. In addition to the importance of frames in representing objectives as discussed in Section 1 frames facilitate meaningful simplification and homomorphism relations the modeller seeks to establish.[3,6] Also, the frame allows for a clear separation of model and experimentation specifications. This results in modular simulation software designs. Such designs incorporate separate modules for model, experiment and execution control specifications. This conceptual framework has, in fact, been realized in some simulation software systems.[6,8,11-13] Finally, the hierarchical frame specification discussed in this paper consolidates efforts to provide a unified framework for simulation of distributed, hierarchically specified systems.

The set of experimentation circumstances defined by a frame consists of input, output, run control and summary variable sets. Constraints are imposed on the time segments of input and run control variables. Formally, the *experimental frame* is the following structure:

$$EF = \langle T, I, O, C, \Omega_I, \Omega_C, SU \rangle$$

where:

- T is the time base
- I is the set of input variables which will be simulated in any model that accommodates the frame, i.e., to which the frame is applicable
- O is the set of output variables which will be observed in a frame applicable model
- C is the set of run control variables which will be monitored but are defined for experimentation control rather than output behavior observation
- Ω_I is the set of input segments, i.e., the allowable sequences (time trajectories) of inputs that will be sent to the model
- Ω_C is the set of run control segments, i.e., constraints on the combinations of run control variables (including temporal constraints) which capture the domain operation required by the frame. Input/Output behavior of a model in this frame is accepted as long as the run control constraints are not violated.
- SU is the set of summary mappings which are statistical and other aggregations of the input/output behavior into reduced and manageable spaces.

Experimental frames are given concrete form as illustrated in Figure 1. Employing the concepts of automata theory and the DEVS (Discrete EVent System specification) formalism, Zeigler[3] defines a *generator* which produces the input segments sent to a model, an *acceptor*, i.e., a device that continually tests the run control segments for satisfaction of the given constraints, and a *transducer* which collects the input/output data and computes the summary mappings.

The DEVS formalism provides a means of specifying a mathematical object called a *system*. Basically, a system has a time base, inputs, states, and outputs, and functions for determining next states and outputs given current states and inputs.[3,6] The insight provided by the DEVS formalism is in the simple way that it characterizes how discrete event simulation languages specify discrete event system parameters. Having this abstraction, it is possible to design new simulation languages with sound semantics that are easier to understand.

DEVS-Scheme,[6,11,13] an implementation of the DEVS formalism in Scheme (a Lisp dialect), supports building models in a hierarchical, modular manner. This is a systems oriented approach not possible in popular commercial simulation languages such as Simscript, Simula, GASP, SLAM and Siman (all of which are discrete event based) or CSMP and ACSL (which are for continuous models). In the DEVS formalism, one must specify 1) basic models from which larger ones are built, and 2) how these models are connected together in hierarchical fashion. In this formalism, basic models are defined by the structure:

$$M = \langle X, S, Y, \delta, \lambda, ta \rangle$$

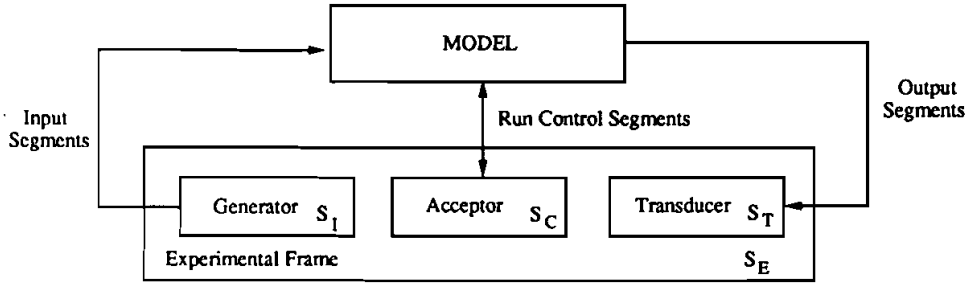

Figure 1 Experimental Frame Structure

where X is the set of external input event types, S is the sequential state set, Y is the set of external event types generated as output, $\delta_{int}(\delta_{ext})$ is the internal (external) transition function dictating state transitions due to internal (external input) events, λ is the output function generating external events at the output, and ta is the time-advance function. Rather than reproduce the full mathematical definition here, we proceed to describe it informally.

To specify modular discrete event models requires that we adopt a different view than that fostered by traditional simulation languages. As with modular specification in general, we must view a model as possessing input and output ports through which all interaction with the environment is mediated. In the discrete event case, events determine values appearing on such ports. More specifically, when external events, arising outside the model, are received on its input ports, the model description must determine how it responds to them. Also, internal events arising within the model change its state, as well as manifest themselves as events on the output ports to be transmitted to other model components. A multi-component model can be expressed as an equivalent basic model in the DEVS formalism.[3,6] Such a basic model can itself be employed in a larger multi-component model. This shows that the formalism is closed under coupling as required for hierarchical model construction.

An experimental frame, E, is realized by a system S_E which is a composition of systems S_I (an (DEVS) input segment generator), S_C (a (DEVS) run control segments acceptor) and S_T (a composition of (DEVS) transducers, each of which realizes a summary mapping). This realization is depicted in Figure 1. Notice that the system S_E is coupled to the model of the system under study. Basic DEVS devices for standard operations, e.g., computation of the average of values, acceptance of a constant segments, can be used to build up more complex frames.

An experimental frame employed in a simulation study applies to a specific model and answers the questions directly addressed to that model. In other words, the variables, segments and constraint conditions apply to the specific model under study. It is thus desirable that frame realizations be a concretization of a general experimental frame type. The general frame type can be regarded as a generic experimental frame for certain classes of problems and types of performance evaluation criteria. Such standard criteria would include input/output performance indexes, utilization of resources measures, reliability assessments, etc. The ensuing section presents the generic frame concept.

2.1 Generic Experimental Frame

A generic frame is a general class of experimental frames from which frames for a specific simulation study can be derived. A generic frame contains generic variables

that correspond to the objective for which the simulation is undertaken; it is a template whose instantiation takes place in the context of the model to be simulated.

With a simulation objective, we associate a performance index that allows for a final judgment of the simulation model with respect to that objective. A *generic experimental frame* is defined as the following structure induced by a performance index *pi*:

$$GEF_{pi} = \langle IG, OG, \Omega_{GI}, SU \rangle$$

where:

GEF_{pi}	denotes the generic experimental frame for performance index *pi*
IG	is the set of generic input variable types for *pi*
OG	is the set of generic output variable types for *pi*
Ω_{GI}	is the set of generic input segment types for *pi*
SU	is the set of summary variables for *pi*

Notice that the set of run control variables and segments is not included in the above definition. The execution control conditions for a simulation run should be specified after the relevant generic frame has been instantiated and the model is ready for experimentation.

2.2 *Example of Experimental Frame Specification*

To illustrate a generic frame specification and its instantiation consider the following example: In many classes of problems one of the standard simulation objectives is to obtain measurements concerning utilization of a system's components. The measure associated with this objective is often called *utilization* and is expressed as follows:

$$utilization = (total\ time\ a\ component\ is\ active / total\ observation\ time).$$

We define the generic frame that reflects this measure. To record the utilization of a component we must monitor its status, i.e., whether it is active or idle. In addition, input variables and segments must be defined in order to observe how the component responds to a sequence of tasks arriving at the system. Thus, the frame is specified as follows:

Generic Frame Type: UTILIZATION
{comment: specifies a class of experimental frames for evaluation of component utilization in discrete event systems}

Generic Input Variables: Arrival with range {0,1}
where 1 denotes an event of arrival, and
0 is an empty event

Generic Output Variables: Status with range {0,1}
where Status = 0 denotes the idle state, and
Status = 1 denotes the active state
Generic Input Segment Type: InputSegment.Arrival
{class: DEVS segment
parameters: inter-arrival distribution type}

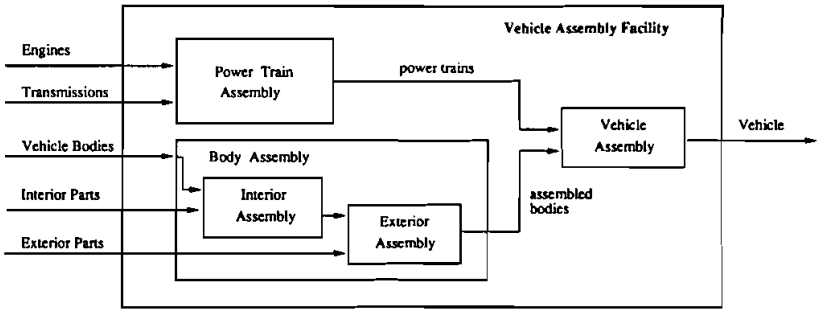

Figure 2 Structure of the Vehicle Assembly Facility

{comment: after the generic frame is instantiated, the segment's description is matched with the specification of standard experimental frame input generators, so that the input segment can be realized as a DEVS generator}

Generic Summary Variable: Utilization

{comment: to obtain values for Utilization, a standard DEVS transducer should be employed. Such a transducer will monitor the variable Status and record the following ratio:

Time(Status = 1)/Total.Elapsed.Time}

Software assistance for the retrieval, definition and modification of generic experimental frames should be provided in advanced simulation environments. Simulation objectives should be represented by a high level description of the associated performance indexes. Based on this description and the knowledge of available generic variables, a generic frame configurator should either locate a relevant generic template in the frame base or construct a new template and store it in the base. In case a template cannot be located or configured the modeller should have the option to add new variables or to create new templates. The realization of a generic frame configurator represents a potential for the use of AI techniques. Rozenblit and Hu explore this potential.[14]

To illustrate how a generic frame is derived from its generic counterpart we employ a model of an assembly system. We shall return to this example throughout the remainder of the paper in order to illustrate the conceptual framework.

Description of the Model

Suppose that a model of an assembly system is given and is to be simulated. As shown in Figure 2, the Vehicle Assembly Facility (VAF) system consists of three assembly clusters (workcells): Power Train Assembly, Body Assembly, and Vehicle Assembly. The Body Assembly module has two submodules, i.e., the Exterior and Interior assemblies. Assume further, that one of the simulation objectives is to evaluate the utilization of the Power Train Assembly workcell.

The inputs to VAF are: *engines* and *transmissions* which are received at the Power Train Assembly; *vehicle bodies* received by the Body Assembly cluster; *interior* and *exterior parts* received by the Interior and Exterior assemblies, respectively. The Vehicle Assembly workcell processes *power trains* and *assembled bodies*, and outputs *assembled vehicles*.

For illustration, an instance of the generic experimental frame UTILIZATION is created for the Power Train Assembly model. The instantiation proceeds as follows: The variables of the generic frame are qualified with the names of the model components to which they apply. Appropriate model variables are selected to serve as the run control variables. The summary variables are again qualified by the names of the model components for which the frame is to gather data. In our example, the experimental frame takes the following format:

Experimental Frame: POWER TRAIN ASSEMBLY UTILIZATION

Input Variables: POWER-TRAIN-ARRIVAL with range {0, 1}

Input Segments: Power-Train-Arrival
{discrete event segment, inter-arrival distribution: exponential
mean inter-arrival time: 1}

Output Variables: POWER-TRAIN-ASSEMBLY-STATUS with range {0, 1}

Control Variables: POWER-TRAIN-ASSEMBLY-QUEUE-LENGTH
with range {non-negative integers}

Control Segments: QUEUE-LENGTH-MONITOR:
run simulation as long as
POWER-TRAIN-ASSEMBLY-QUEUE-LENGTH < 5

Summary Variables: POWER-TRAIN-ASSEMBLY-UTILIZATION
with range [0, 1] (computed as given in the generic frame UTILIZATION)

This frame can be realized by a DEVS-generator, acceptor, and transducer, as discussed in Section 2. Further examples of formal, DEVS-based experimental frame definitions are given in.[2,15]

The steps of the experimental frame specification framework discussed so far are:

1. An evaluation objective underlies the definition of its corresponding generic experimental frame
2. Given a model which is to be evaluated in this objective, an instance of the generic frame is created.
3. This instance is an experimental frame that can be realized as a composition of generator, acceptor and transducer modules

In presenting and demonstrating the frame concepts, we have not taken into account the multiplicity and hierarchy of models in large scale systems simulations. The next section addresses this issue.

3. A FRAMEWORK FOR HIERARCHICAL, DISTRIBUTED EXPERIMENT SPECIFICATION

This section extends the definition of experimental frame and discusses a framework for frame specification in a distributed DEVS simulation environment. So far, we have coupled models and experimental frames only at the highest level, that is, without taking into account the hierarchical structure of the model. Now, an approach

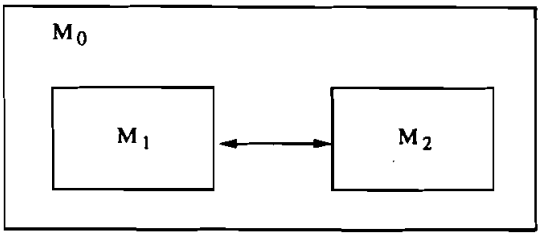

Figure 3 Multicomponent Model Structure

to distributing an experimental frame within a hierarchical model is presented which facilitates attachments of frames to both atomic and coupled subcomponents of a hierarchical model. Distributing frames in this manner offers a means of exploiting parallelism and modularity in distributed simulation. The DEVS abstract simulator[3] is the basis of our considerations. We assume that an experimental frame is expressed in the DEVS form. We shall define appropriate model/frame coupling mechanisms. A prototype implementation of these concepts has been built in DEVS-Scheme.[15]

3.1 Hierarchical Specification of Experimental Frames

The simulation of DEVS models is based upon the abstract simulator developed as a part of the DEVS theory.[3] The abstract simulator concepts are implemented in DEVS-Scheme by three specialized classes of processors: Simulators, Co-ordinators, and Root-co-ordinators. The root-co-ordinator is the manager of the overall simulation process and is linked to the co-ordinator of the highest level coupled-model. Simulators and Co-ordinators are used to handle the atomic-models and coupled-models respectively. The simulation process is managed by passing messages between the specialized processors. The messages carry internal event, external event, and synchronization information.

We now illustrate the *abstract* simulator concept in more detail. Assume that a multicomponent model M_0 as presented in Figure 3 is expressed in DEVS formalism. An abstract simulator of M_0 takes the form depicted in Figure 4, where both S_1 and S_2 are simulators and C_0 is a coordinator. The simulators S_1 and S_2 interpret the dynamics of model components M_1 and M_2, respectively. The coupling of S_1, S_2 and

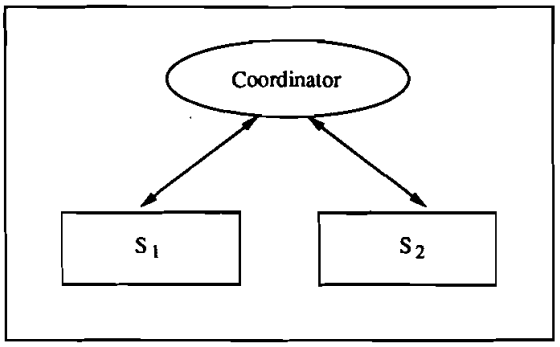

Figure 4 Abstract Simulator for Model M_0

the coordinator C_0 is itself an abstract simulator that simulates model M_0. As we can notice, a one-to-one correspondence between the structure of a model and that of the simulator exists. We now characterize the principles underlying the operation of the abstract simulator and coordinator. The reader is referred to[3,6] for further details.

The operation of an abstract simulator involves handling four types of messages: $(*, t)$, (x, t), (o, t), and (y, t). In each case, the right hand element is the global clock time of the simulated DEVS. When the simulator receives a $(*, t)$ message, it undergoes its internal state transition and sends a (y, t) message to its coordinator as an output. When it receives an (x, t) message, it undergoes external event-generated transition. The message (o, t) causes the simulator to send its output as a (y, t) message to its coordinator.

A coordinator carries out its task by mediating three types of messages sent to and from the parent coordinator; denoted $(*, t)$, (x, t) and (o, t), where the right hand element is the global clock time of the simulated DEVS. The $(*, t)$ message indicates that the node should be activated, i.e., an internal event should be executed in the DEVS at the node. When a $(*, t)$ message is received by a coordinator, it is transmitted to the sub-ordinate representing the imminent component DEVS. When $(*, t)$ is received by a leaf simulator, it carries out the internal transition function of the associated DEVS. Upon receipt of a $(*, t)$, a coordinator also transmits (o, t) messages to each of its subordinates requesting that each return the output corresponding to its associated DEVS. (Transfers of $(*, t)$ and (o, t) messages are depicted in Figure 5.)

Finally, the (x, t) message indicates that an external event x is arriving at the global time t. When received by a coordinator, it consults its external-to-internal coupling table to generate appropriate (x, t) messages to the subordinates influenced by the external event. When (x, t) is received by a leaf simulator, it directly executes the external transition of the associated DEVS.

Although, an (x, t) message may originate from the environment external to the overall model, it may also be generated within the hierarchy. The latter occurs when the outputs received by an activated coordinator in response to its (o, t) request are collected together using its internal-to-external coupling table. The resulting (y, t) message is sent to the parent coordinator for distribution as (x, t) messages to the subordinates influenced by the activated coordinator. Again, we emphasize that the abstract simulators are essential to our framework since they execute both the model and frame components specified in the DEVS description.

For the purpose of illustration, assume that a model consist of two atomic submodels as illustrated in Figure 3. The abstract simulator for such a model has the structure depicted in Figure 4. We now describe how an experimental frame module can be synthesized and coupled with the model's abstract simulator. Two types of architectures are discussed: 1) a centralized experimentation mode, and 2) a distributed frame architecture.

3.2 Centralized Experimentation

The basic experimental frame/model coupling results in the architecture depicted in Figure 6. This coupling is described below.

Recall from the definition of the experimental frame realization that each component of the system S_E (Figure 6) (i.e., generator, acceptor and transducer) is a DEVS model and thus may be realized as a hierarchical coupling of DEVS specified systems. In the centralized architecture illustrated in Figure 6 control is concentrated

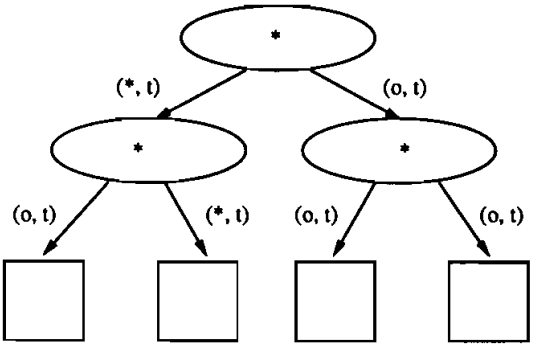

Figure 5 Propagation of (*, t) and (o, t) messages

within the master experimental frame module S_E whereas the simulators S_1, S_2 are responsible for execution of model component's dynamics.

The coupling of the frame module S_E and the abstract model simulator S is defined as follows: the generator S_G originates the messages (x, t) that are received by the root coordinator C_0 as external events to the model. The output statistics are gathered by the transducer S_T that collects the (y, t) message from the root coordinator. This message defines an input signal to the frame transducer S_T. It carries the information about changes of output variables in each subordinate DEVS model simulator.

The realization of experimentation control requires that the coordinator of each abstract simulator be extended as follows: upon receipt of a (*, t) or an (x, t) signal, the coordinator transmits (m (monitor), t) messages to its subordinates requesting that each return the message (c (control), t) corresponding to a change (if any) of the control variables' values of an associated DEVS model component. The global message (c, t) is collected by the root coordinator and processed by the frame acceptor S_A which determines whether or not the run control segments lie within the admissible range.

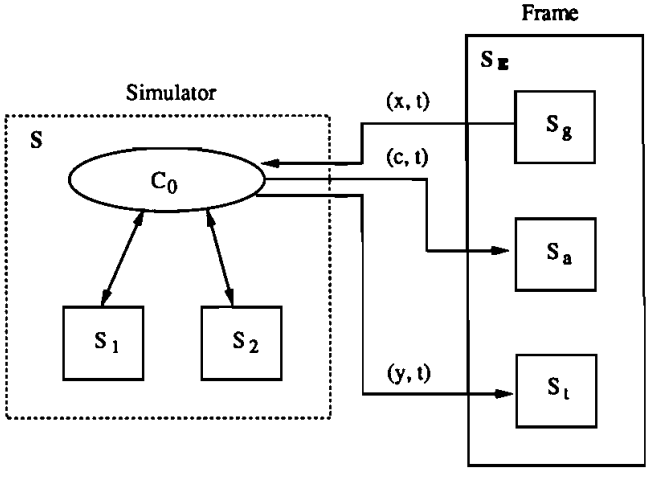

Figure 6 Centralized Experimental Frame

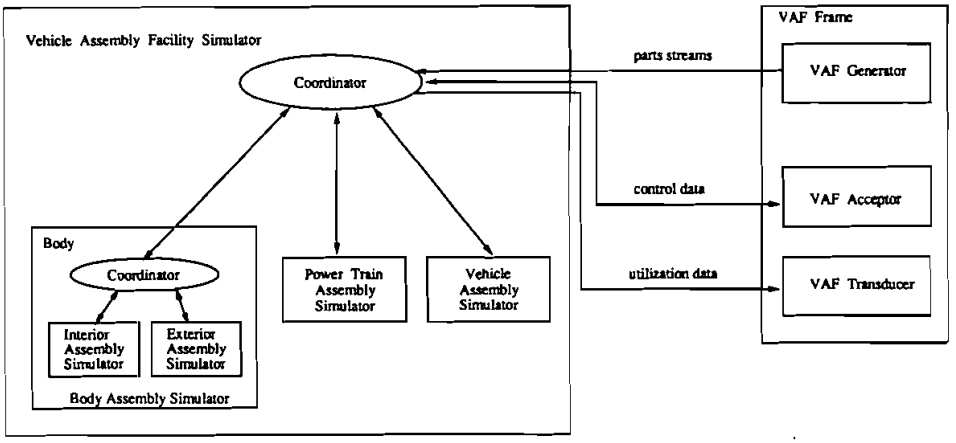

Figure 7 Vehicle Assembly Facility Simulator with Centralized Experimental Frame

3.2.1 Example of centralized experimental frame Recall the Vehicle Assembly Facility described in Section 2.2. Figure 7 depicts the structure of the abstract simulator of this facility with a global experimental frame (VAF Frame) attached to it. Assume that the simulation objective is to evaluate the utilization of workcells in the VAF. More specifically, the following data are sought: 1) utilization of each individual workcell, 2) average utilization and joint utilization of the Body Assembly Cluster, 3) average and joint utilization of VAF. The *average utilization* is computed as the average of the utilizations of each component in a cluster whereas the *joint utilization* is the ratio of time during which all workcells are simultaneously in service versus total observation time, i.e., *joint-utilization* = *total-time*(*Status*($workcell_1$) = *1 and Status*($workcell_2$) = *1 and . . . and Status*($workcell_n$) = *1*)/*total-observation-time*

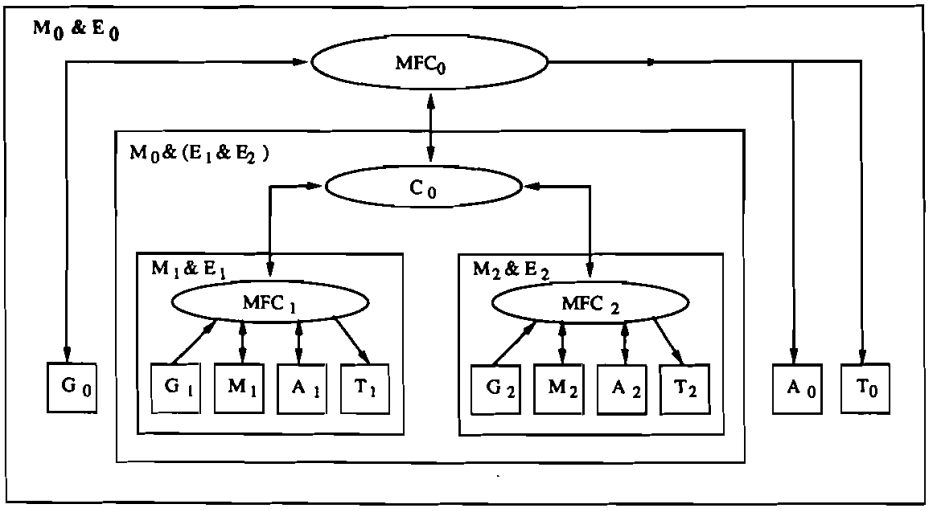

Figure 8 Distributed Experimental Frame Architecture for Distributed Model

for workcells $i = 1 \ldots n$. The joint utilization indicates the degree of concurrency of processes in the system and may be used in planning and scheduling service at workcells, load balancing, etc.

The VAF experimental frame must be defined in such a way that the above listed measures can be computed. (For the sake of brevity, we limit the description to functional specifications for each VAF component. Also, we assume that the only run control condition is the length of the simulation experiment). The frame's components are:

VAF Global Frame

VAF Generator: generates input segments *VAF-engine-arrival, VAF-transmission-arrival, VAF-vehicle-body-arrival, VAF-exterior-parts-arrival, VAF-interior-parts-arrival.*

Comment: each segment is a discrete event segment with a certain interarrival distribution. The arrival events are sent to the root coordinator that routes them accordingly.

VAF Acceptor: monitors the simulation clock. Simulation is run while Clock < T-End.

VAF Transducer: observes status of each workcell, i.e., Interior.Assembly.Status, Exterior.Assembly.Status, Power.Train.Assembly. Status, Vehicle.Assembly.Status, and computes the utilization of each workcell as prescribed in the frame of Section 2.2.

Comment: values of each Status variable must be passed on to the frame by the root coordinator which receives them from the lower level coordinator and component simulators.

The Body.Assembly.Average.Utilization is computed as the average of Interior.Assembly.Utilization and Exterior.Assembly.Utilization. VAF. Average. Utilization is computed for the entire facility as the average of the workcells' utilizations.

The Body.Assembly.Joint.Utilization is computed by observing the status of this cluster's sub-assemblies and keeping track of the time both sub-assemblies are simultaneously in service. Then, this time is divided by the total observation time. The joint utilization of the entire system is computed analogously.

The centralized architecture involves a single experimental frame module directly linked to the global coordinator. Each frame component (i.e., generator, acceptor, and transducer) may in general prove very complicated due to the complexity of the functions it executes (e.g., the above transducer). Alternatively, the components of experimental frames be distributed in a manner that corresponds to the hierarchical, distributed structure of the models they are applicable to. This requirement has been stipulated in the literature by Dekker[16] (the concept of a cosystem), Oren[9] (GEST implementation of local frame segments) and Biles[7] (distributed evaluation of a network of microprocessors).

3.3 Distributed Experimental Frame Architecture

In order to specify a distributed experimental frame, we establish the scheme for its top-down decomposition. First, consider the input generation process. Assume that

at any given level of the hierarchy of model decomposition the model M_i has constituent models $M_{i,1}, \ldots, M_{i,k}$. In the centralized mode of experimentation, a generator for this model, G_i has to be defined and coupled to M_i through its input ports. In order to realize G_i as a coupling of subcomponent—possibly less complex—generators, we have to identify the structure of the input segments received by the model M_i. In the most general case, we can assume that an input segment is decomposed into mutually independent segments $\omega_{i,1}, \ldots, \omega_{i,k}$ that are applied directly to model components $M_{i,1}$ through $M_{i,k}$ and the segment $\omega_{i,0}$ which accounts for input to their coupling, i.e., M_i. In other words, G_i generates segments $\omega = (\omega_{i,0}, \omega_{i,1}, \ldots, \omega_{i,k})$. We decompose G_i into generators $G_{i,0}, G_{i,1}, \ldots, G_{i,k}$, and couple them with their respective model simulators. The coupling is accomplished by a parallel composition of DEVS-specified models that realize the generators $G_{i,0}, G_{i,1}, \ldots, G_{i,k}$. The parallel coupling of component generators is a DEVS in a modular form in which no component is an influencee of another component.

Notice that any model component may itself be composed of submodels. Then, its corresponding generator is decomposed in the manner described above. Such a process is carried out recursively down to the leaf nodes of the model composition tree (hierarchy of components' decompositions).

The decomposition process of the output transducer is similar to that of the input generator. The transducer T_i collects global output segments $\rho = (\rho_{i,0}, \rho_{i,1}, \ldots, \rho_{i,k})$ where $\rho_{i,0}$ may represent correlated output of the components $M_{i,1}, \ldots, M_{i,k}$ while $\rho_{i,1}, \ldots, \rho_{i,k}$, are mutually independent, local output segments. We carry out the decomposition of the output transducer as follows: T_i is decomposed into $T_{i,0}, T_{i,1}, \ldots, T_{i,k}$ that are coupled to their respective model components $M_i, M_{i,1}, \ldots, M_{i,k}$.

Notice that the above specification establishes frames at any two subsequent levels of the model composition tree and that the process of associating transducers with model components can be carried out recursively down to the leaf nodes of the tree.

The run control acceptor A for the model M_i is decomposed in exactly the same way as the output transducer. The component acceptors $A_{i,0}, A_{i,k}, \ldots, A_{i,k}$ monitor the run control trajectories $c_{i,0}, c_{i,1}, \ldots, c_{i,k}$, respectively. Conceptually, $A_{i,0}$ checks for acceptance of the global run control segment pertaining to M_i while the component acceptors monitor the control segments local to $M_{i,k}, \ldots, M_{i,k}$. Once again, this establishes the specification framework for any two subsequent levels of the composition tree and this process is recursive with respect to the number of levels in the tree. The hierarchical specification of the run control acceptor is analogous to the specification of the transducer.

We proceed to describe how an experimental frame is mapped onto a distributed architecture of a DEVS simulator.

3.4 Mapping Hierarchical Specification of Experimental Frames onto the DEVS Abstract Simulator

The design of a methodology for mapping the decentralized frame specification onto the corresponding abstract distributed simulator should satisfy the following requirements:

- The coupling of the simulator and frame must be closed, i.e., it must result in an abstract simulator.

- The degree of decentralization of experimentation should be maximal. In other words, a means of assigning an experimental frame local to each model component should be provided.

Motivated by the above guidelines, we suggest the following procedure for establishing the frame/abstract simulator mapping: At the level (i) of the model composition tree, a DEVS simulator of a model component must simulate the model with a pertinent experimental frame. Recall that the frame components are defined as DEVS systems and thus can be realized by an abstract simulator as well. However, coordination is required between the simulators of the model, generator, acceptor and transducer. To assure such coordination, we introduce a model/frame coupler (MFC).

An MFC is a coordinator whose functions are: At the level local to its frame and model (i.e., the level (i)), the MFC sends the (*, t) message to the frame generator. This message results in an internal transition of the generator and a message (y, t) being output by the generator. This (y, t) message is sent back to the model/frame coupler and forwarded directly as an external event (x, t) to the simulator of the model component. The MFC also forwards a local (y, t) message generated by the model simulator to the local frame transducer and a (c, t) message to the local acceptor, respectively. Notice that both the transducer and acceptor are passive DEVS systems (i.e., they cannot activate themselves). This significantly simplifies the design of the MFC since it has to schedule only the internal transitions of one active component i.e., the generator. The coupler also serves as a communication port with the parent coordinator specified at level ($i - 1$) of the simulator hierarchy. Its function as an i/o port consists in transducing the (*, t), (x, t), (o, t), (m, t) and (c, t) signals to(from) the parent coordinator from(to) the simulator of the model component at the subordinate level.

To exemplify the discussion let us consider the simulator presented in Figure 4. (Figure 8 shows a fully distributed frame structure coupled with this simulator). The coupler MFC_1 coordinates the simulator of the model component M_1 and corresponding simulators of G_1, T_1 and A_1. It broadcasts messages (*, t) to the generator which responds by producing an output signal (y, t). This output signal is in turn transduced by MFC_1 to the simulator M_1. The coupler collects the messages (y, t) and (c, t) from M_1 and transduces them to T_1 and A_1, respectively. The composition of MFC_1, M_1, G_1, T_1 and A_1 constitutes the simulator for the component M_1 with its corresponding experimental frame E_1, denoted as M_1 & E_1. The simulator M_2 & E_2 is realized in the same manner. Both simulators are coupled by the standard (in the sense of Zeigler's definition[3]) coordinator C_0. The role of MFCs in the coupling is restricted to serving as input/output ports to the combined model/frame simulators. They simply transduce the messages between C_0 and M_1 and M_2. Notice, however, that although C_0 is the root coordinator, it is still necessary to simulate the model M_0 and its frame E_0. To achieve this, MFC_0 is created to coordinate the actions of the simulators M_0, G_0, T_0 and A_0. This model/frame coupler is linked to the root coordinator. The experimental frame/model coupling resulting at this level is M_0 & E_0 as shown in Figure 8. The model/frame coupler transmits the generator's outputs as external event messages to the coordinator C_0. It also receives the global (y, t) output and (c, t) control messages. These messages are sent to the transducer and acceptor, respectively.

3.4.1 Example of distributed frame architecture Again, we return to the Vehicle Assembly Facility example and show how the centralized experimental frame given

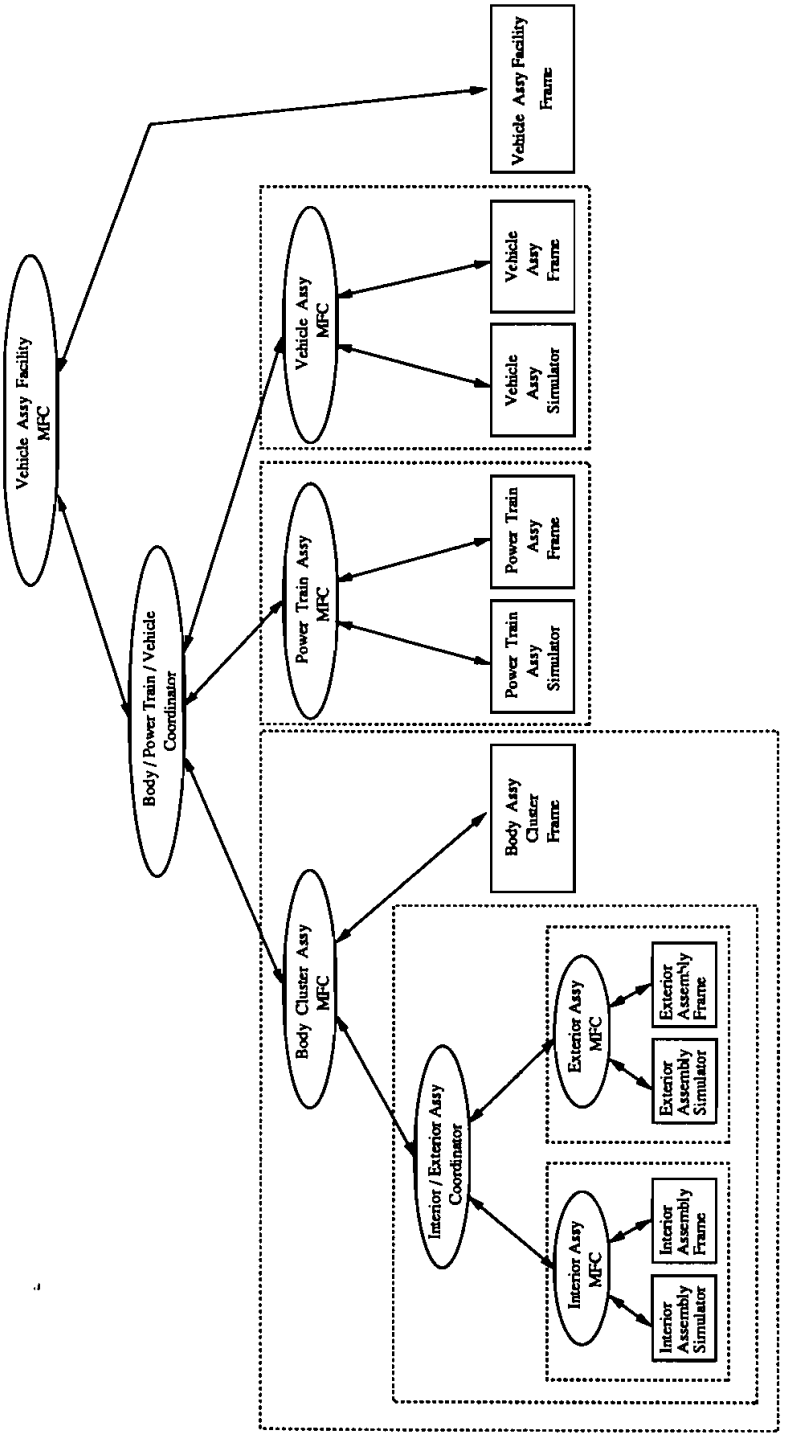

Figure 9 Vehicle Assembly Facility Simulator with Distributed Experimental Frame

in Section 3.2.1 can be distributed using the mapping scheme just discussed. In the ensuing discussion, the evaluation objective is once more the utilization of components.

Figure 9 shows the VAF with experimental frames attached at each level of the simulator's hierarchy. We proceed with a detailed description in the bottom-up manner.

Interior Assembly Frame

Interior assembly generator generates the input segment Interior-Assembly-part-arrival (discrete event segment with an inter-arrival distribution)

Interior assembly acceptor none

Interior assembly transducer computes Interior.Assembly.Utilization (as described in Section 2.2)

The Interior Assembly Frame is a local frame with respect to the Interior Assembly Simulator to which it is attached through the Interior Assembly MFC.

Exterior Assembly Frame

Exterior assembly generator generates the input segment Exterior-Assembly-part-arrival (discrete even segment with inter-arrival distribution)

Exterior assembly acceptor none

Exterior assembly transducer computes Exterior.Assembly.Utilization

The Exterior Assembly Frame is also a local frame with respect to the Exterior Assembly Simulator to which it is attached through the Exterior Assembly MFC.

Body Assembly Cluster Frame

Body assembly cluster generator generates the input segment Vehicle-body-arrival

Notice that this segment is an input to the coupled level model of two subcomponents. This is consistent with our decomposition of a generator, discussed in Section 3.3.

Body assembly cluster acceptor none

Body assembly cluster transducer computes Average.Body.Cluster.Assembly.Utilization and Joint.Body.Cluster.Assembly.Utilization.

The Body Assembly Cluster Frame is considered global with respect to the Interior and Exterior sub-assemblies and local with respect to the Body Assembly Cluster Simulator to which it is attached through the Body Assembly Cluster MFC.

Power Train Assembly Frame

Power train assembly generator generates the input segments Power-Train-Assembly-transmission-arrival and Power-Train-Assembly-engine-arrival.

Power train assembly acceptor none

Power train assembly transducer computes Power Train.Assembly.Utilization

The Power Train Assembly Frame is local with respect to the Power Train Assembly Simulator to which it is attached through the Power Train Assembly MFC.

Vehicle Assembly Frame

Vehicle assembly generator none (parts are received from the Power Train and Body assemblies)

Vehicle assembly acceptor none

Vehicle assembly transducer computes Vehicle.Assembly.Utilization.

The Vehicle Assembly Frame is also local with respect to the Vehicle Assembly Simulator to which it is attached through the Vehicle Assembly Model Frame Coupler (MFC).

The Body Cluster, Power Train, and Vehicle assemblies, each with its corresponding local frame, are coordinated by the Body/Power Train/Vehicle Coordinator. The highest level frame is attached through the Vehicle Assembly Facility MFC to this coordinator. The frame is defined functionally as follows:

VAF Frame

VAF assembly generator none (parts arrive at subassemblies from the local generators)

VAF assembly acceptor monitors the simulation time and terminates the execution if the time exceeds a certain limit.

VAF assembly transducer computes Average.VAF.Utilization and Joint.VAF.Utilization.

To ensure consistency in attaching a frame to a simulator, we verify that the frame-simulator/model-simulator coupling relations are valid. More specifically, the output ports produced by the generator must match the input ports of the model, the output ports of the model must match the input ports of the transducer, and the variables monitored by the acceptor must match those designated as the run control variables. The MFC module checks if the above requirements are met.

The proposed mapping results in an abstract simulator that simulates the combined model/frame DEVS. Since each experimental frame module is a special form of a DEVS simulator, i.e., a DEVS-generator, acceptor, and transducer, the coupling of frame and model simulators results in a DEVS simulator. The correctness of simulation is then ensured by the correctness of the DEVS simulator (for a formal proof of DEVS simulator correctness see Ref. 3).

Since a means for coupling an experimental frame to a model component at any level of the hierarchy are provided, it is apparent that maximum decentralization of the experimentation can be achieved. The significance of the frame distribution scheme is that it facilitates inspection of the model's behavior at each level of abstraction. We explore this issue in the next section.

4. IMPACT ON DESIGN OF AUTONOMOUS SYSTEMS

In principle, an autonomous system could base its operation on a comprehensive model of its environment (and itself). However, to develop such a model would be an intractable task. Instead, Zeigler[17] proposes a model-based architecture for autonomous systems in which a multiplicity of partial models of different levels of abstraction are employed. The partial models are oriented towards specific objectives, and thus need to be evaluated in respective experimental frames that reflect those objectives.

The experimental frame specification methodology presented here integrates well with the model-based approach to high autonomy systems. First, it provides a systematic approach to defining a set of conditions under which an autonomous system is to operate and for measuring the degree to which the system is capable of performing certain actions. The ability to achieve a pre-specified objective can be tested within a frame that defines this objective. The ability to adapt to major environment changes can be measured by emulating the changes through an appropriate experimental frame. Testing the ability of the system to develop its own objectives will not only involve the creation of new models to support the new objectives but also the development of relevant experimental frames that reflect those objectives.

Secondly, the distributed frame architecture supports flexible experimentation with multicomponent systems that may exhibit various degrees of distribution and coordination among their components. It is the level of abstraction at which we wish to observe the behavior of the system that determines where we attach the frame components and how we define their functions. Thus, the degree of autonomy of individual system components may be observed in local frames that pertain to those components or within higher level frames that assess the coordination/cooperation among the components. Imagine that the Vehicle Assembly Facility is serviced by robots which process details at the workcells. One possible model of the VAF could treat the Power Train Assembly, the Body Assembly Cluster, and the Vehicle Assembly workcells as entirely independent units. Thus only local frames would be defined for these units. However, within the Body Assembly Cluster a higher level frame (in addition to two local frames that govern the evaluation of the Interior and Exterior assemblies) may be defined to measure the degree of cooperation among robots processing the vehicle body. Here, we have assigned a higher level frame to a cluster based on the strong coupling between the cluster's subsystems, i.e., furnishing the body with interior and exterior parts are closely related operations with a common platform—the vehicle body. In future research, criteria for frame assignments will have to be developed in model-based architectures for autonomous systems.

Finally, isolation of model components for testing and validation can be easily performed as discussed in Refs. 15,18–20.

5. THE METHODOLOGY REVISITED—SUMMARY

The steps of the experimental frame specification methodology are summarized below:

1. An evaluation objective underlies the definition of its corresponding generic experimental frame

2. Given a model which is to be evaluated in this objective, an instance of the generic frame is created.
3. This instance is an experimental frame that can be realized as a composition of the generator, acceptor and transducer modules.
4. Large scale system simulation involves multicomponent models. The specification of experimental frames for such models may take two forms:
   - the centralized experimental frame is a global frame module connected to the model of the entire system. The realization of such a frame module may prove complex if the observation conditions are intricate.
   - the distributed frame architecture facilitates the attachments of experimental frames to models at different levels of the system's decomposition hierarchy. At a given level, a frame is local with respect to the model components specified at this level. At the next higher (coupled system) level, a higher level frame that addresses the evaluation issues for the coupled model can be specified.

The utility of the proposed framework in the context of distributed simulation and model-based architectures for high autonomy systems has also been discussed. Future research will attempt to establish criteria for experimental frame specification in autonomous systems simulation.

REFERENCES

1. J. W. Rozenblit, "Experimental Frames for Distributed Simulation Architectures." *Proceedings of the 1985 SCS Multiconference, Distributed Simulation,* San Diego, California, January 1985, pp. 14–20.
2. J. W. Rozenblit, "A Conceptual Basis for Integrated, Model-Based System Design." Ph.D. Dissertation, Department of Computer Science, Wayne State University, Detroit, Michigan, 1985.
3. B. P. Zeigler, *Multifacetted Modelling and Discrete Event Simulation.* Academic Press, 1984.
4. W. Jacak and J. W. Rozenblit, "Automatic Simulation of a Task-Oriented Robot Program for a Manufacturing Process." *Robotica* (to appear), 1991.
5. A. Kusiak and G. Finke, "Selection of Process Plans in Automated Manufacturing Systems." *IEEE Trans. on Robotics and Automation,* **4,** No. 4, 1988, pp. 397–408.
6. B. P. Zeigler, *Object-Oriented Simulation with Hierarchical Modular Models: Intelligent Agents and Endomorphic Systems.* Academic Press, 1990.
7. W. Biles et. al., "Statistical Conosiderations in Simulation of a Network of Microcomputers." *Proc. of the 1985 Winter Simulation Conference,* San Francisco, California, 1985, pp. 388–393.
8. A. Javor, "Demon Controlled Simulation for Efficient Problem Solving." *Proceedings of the 1990 IMACS European Simulation Meeting,* Esztergom, Hungary, 1990, pp. 25–34.
9. T. I. Ören, "GEST—A Modelling and Simulation Language based on System Theoretic Concepts." In: *Simulation and Model-Based Methodologies: An Integrative View,* edited by Ören, T. I., Zeigler, B. P. and M. S. Elzas, North Holland, 1984, pp. 3–40.
10. M. D. Mesarovic and Y. Takahara, *General Systems Theory: Mathematical Foundations.* Academic Press, 1975.
11. T. G. Kim and B. P. Zeigler, "The DEVS-Scheme Simulation and Modelling Environment." In: *Knowledge Based Simulation: Methodology and Application,* edited by Paul A. Fishwick and Richard B. Modjeski, Springer Verlag, 1991.
12. C. D. Pegden, *Siman.* Systems Modeling Corporation, 1982.
13. B. P. Zeigler, "Hierarchical, Modular Discrete-Event Models in an Object-Oriented Environment." *Simulation,* **50,** No. 5, 1987, pp. 219–230.
14. J. W. Rozenblit and J. F. Hu, "Experimental Frame Generation in a Knowledge-Based System Design and Simulation Environment." In: *Modeling and Simulation Methodology: Knowledge System Paradigms,* edited by M. Elzas et al., North Holland, 1989, pp. 451–466.

15. J. Duh, "DEVS-Scheme Implementation of a Distributed Experimental Frame Architecture." Master Thesis, University of Arizona, Tucson, Arizona, 1988.
16. L. Dekker, "Concepts for an Advanced Parallel Architecture." In: *Simulation and Model-Based Methodologies: An Integrative View*, edited by T. I. Ören, B. P. Zeigler, and M. S. Elzas, Springer-Verlag, New York, 1984, pp. 235–280.
17. B. P. Zeigler, "High Autonomy Systems: Concepts and Models." *Proc. of the International Conference on AI, Simulation and Planning in High Autonomy Systems*, IEEE Computer Press, 1990, pp. 136–141.
18. E. R. Christensen, "Hierarchal Optimistic Distributed Simulation: Combining DEVS and Time Warp." Ph.D. Dissertation, University of Arizona, Tucson, Arizona, 1990.
19. R. G. Sargent, "An Exploration of Possibilities for Expert Aids in Model Validation." In: *Modelling and Simulation in the Artificial Intelligence Era*, edited by M. S. Elzas, T. I. Ören and B. P. Zeigler, North-Holland, Amsterdam, 1986, pp. 279–298.
20. B. P. Zeigler, J. W. Rozenblit and E. R. Christensen, "Reducing the Validation Bottleneck with a Knowledge-Based, Distributed Simulation Environment." *Expert Systems with Applications* (to appear), 1990.

Jerzy W. Rozenblit is an Assistant Professor of Electrical and Computer Engineering at The University of Arizona, Tucson. He received the Ph.D. and M.S. degrees in Computer Science from Wayne State University, Detroit, in 1985 and 1983, respectively, and the M.S. degree in Control Engineering from the Technical University of Wroclaw, Poland, in 1980. He specializes in modeling and computer simulation, knowledge-based system design, and artificial intelligence. His principal research activities focus on the development of expert, computer-aided environments for engineering design support.

Dr. Rozenblit has consulted for Martin Marietta, Semiconductor Research Corporation (SRC), and Siemens, and has been a reviewer for the National Science Foundation, Research Council of Canada, and a number of archival journals. He also serves as an Associate Editor of ACM Transactions on Modeling and Computer Simulation. His research in system design has been supported by NASA, Siemens, McDonnell Douglas, SRC, and the National Science Foundation.