Experimental frames for distributed simulation architectures[1]

Jerzy W. Rozenblit
School of Engineering and Computer Science
Oakland University
Rochester, Michigan 48063

ABSTRACT

The paper considers the experimentation aspects of the objectives-driven modelling methodology in the context of distributed simulation. A concept of expressing modelling objectives through the notion of a experimental frame is discussed. A top-down frame decomposition methodology is proposed. Such a methodology results in an abstract distributed simulator architecture that directly implements the principle of model/experimental frame separation (6,10).

INTRODUCTION

Modelling and simulation designates a host of activities associated with constructing models of real world systems and simulating them on a computer. Such activities usually comprise the following stages: system decomposition, model construction, model and experimentation specification. In the paper we focus on the experimentation aspects of modelling methodology. The purposes for which the simulation study is undertaken are operationalized in the process that results in a formal definition of an experimental frame. We employ the DEVS (Discrete Event System) formalism to define the structural representation of an experimental frame as a coupling of a discrete event generator, acceptor and transducer (10). Motivated by the foundations underlying the objectives-driven modelling methodology as defined by Zeigler, we define distributed experimentation paradigms and provide a means of top-down decomposition of experimental frames and mapping of a decomposed frame onto components of a distributed simulator. This in turn leads to decentralization of experimentation control in distributed simulation systems.

OBJECTIVES-DRIVEN MODELLING METHODOLOGY

The conceptual basis for a methodology of model construction in which the objectives of modelling play the key and formally recognized role (therefore called objectives-driven methodology) was laid down by Zeigler (10).

We shall begin with theoretical foundations of the objectives driven methodology. The basic process in such a methodology is that of defining the experimental frame i.e., a set of circumstances under which a model or real system is to be observed and experimented with. This process comprises the following steps. The purposes (objectives) for which the simulation study is undertaken lead to asking specific questions about the system to be simulated. This in turn requires that appropriate variables be defined so a modeller can answer these questions. Ultimately such a choice of variables is reflected in experimental frames which also express constraints on the trajectories of the chosen variables. The constraints on observations and control of an experiment should be in agreement with the modelling objectives. A choice of relevant variables constitutes the first important stage of experimental frame specification. The next step is to categorize the variables into input, output and run control categories and place constraints on the time segments of these variables. Formally, the experimental frame specifies the following seven tuple:

$$EF=<T,I,O,C,\Omega_I,\Omega_C,SU>$$

where
- T is a time base
- I is the set of input variables
- O is the set of output variables
- C is the set of run control variables
- Ω_I is the set of admissible input segments, i.e. a subset of all time segments over the crossproduct of the input variable ranges
- Ω_C is the set of run control segments, i.e. a subset of all time segments over the crossproduct of the control variable ranges.
- SU is a set of summary mappings

The I/O data space defined by the frame is the set of all pairs of I/O segments:

$$D=\{(\omega,\rho) \mid \omega \in (T,X) \ , \ \rho \in (T,Y) \text{ and } dom(\omega)=dom(\rho)\}.$$

where X and Y are input and output value sets, respectively.

The reader is referred to (6,10) for detailed formal definition of an experimental frame. We shall, however, explain the concept of run control variables and segments. In the case of experimentation on a real system, there is no concept of initial state. Thus, specifying the input segment in the frame is not sufficient to determine the output of the system. Since experimental frames should have an interpretation for both the model and the real system, we should provide a meaningful concept of restricting the initial state for the model. The notion of run control variables serves this purpose. Not only do the run control variables initialize the experiments, they also set up the conditions for continuation and termination. The set of initialization conditions constitutes a subset of the control space called INITIAL. Similarly, the subset of the control space defined by the termination conditions is called TERMINAL. These two sets have the following impact on the experimentation. An experiment starts with the control variable values in the INITIAL set and terminates as soon as the TERMINAL subset is entered. In other words, it is continued as long as

[1] This research was supported by NSF grants MCS 8305168, "Theory of Discrete Event Models: Distributed Simulation of Multilevel Models", and NSF grant DCR 8407230, "Distributed Simulation of Hierarchical, Multilevel Models" during the author's tenure with Wayne State University, Detroit, Michigan 48202.

the values of the control variables stay in the subset called CONTINUATION. Thus, we arrive at the definition of the set of run control segments

$$\Omega_C = \{\mu \mid \mu : <t_i, t_f> \text{--->} Z$$

and $\mu(t_i) \in$ INITIAL $\mu(t) \in$ CONTINUATION for $t \in [t_i, t_f)\}$. where Z=crossproduct of the ranges of individual control variables.

STRUCTURAL REALIZATION OF EXPERIMENTAL FRAMES

The concept of realization of any system is concerned with providing the internal structural description of the system given its external behavioral I/O relation.

Zeigler (10) suggests that when realizing the input segments' component of an experimental frame we can engage in experimentation by generation or acceptance. In the case of generation we realize the set of admissible input segments by employing a special class of a DEVS system, called generator. Such a generator starts from a suitable initial state and runs for a desired observation interval. In experimentation by acceptance, we only collect data from the real system that is of interest to us. Thus, the I/O data space determined by the input segments' acceptor is usually a subset of the data space available from the real system. We observe real system's I/O pairs (ω, ρ) and accept such pairs if, and only if, they belong to the data space D defined by the frame.

The appropriate system for realization of run control segments is again the acceptor. An acceptor should be linked to a model in order to monitor the run control segments Ω_C. In the case of Ω_C specified by INITIAL and CONTINUATION subsets of the control space Z, the acceptor can take the following special form. Consider such a device receiving $\mu \in (T,Z)$ as its input segment. In its initial state q_0, it checks whether the initial input value $\mu(0)$ is in INITIAL. If so, it immediately transits to state q_1 and stays there as long as the current input value $\mu(t)$ is in CONTINUATION. Both q_0 and q_1 are acceptance states. If the initial input is not in INITIAL the system transits immediately from q_0 to a dead state q_2, as it does likewise from q_1, if an input value in TERMINAL is received.

For gathering summary statistics we aim to use a transducer. The DEVS transducer is defined as a discrete event system with a designated initial state. When started in such a state it maps its input segments into output trajectories.

Having provided the framework for input/control segments generation and/or acceptance we arrive at the structural realization of an experimental frame as illustrated in Figure 1.

The experimental frame E is realized by the parallel coupling of the DEVS generator S_g, DEVS acceptor S_a and DEVS transducer S_t. This type of realization allows for explicit separation of models and their related frames. The need for such a separation stems from two reasons. First, it is desirable to minimize the effort of model entities to gather data about themselves. Secondly, the model-based simulation methodology suggests the following framework for simulation program development (6,10). In order to support the modular design there should be a model and experimental frame specification modules, and an execution control module responsible for selecting a finite set of all possible experiments

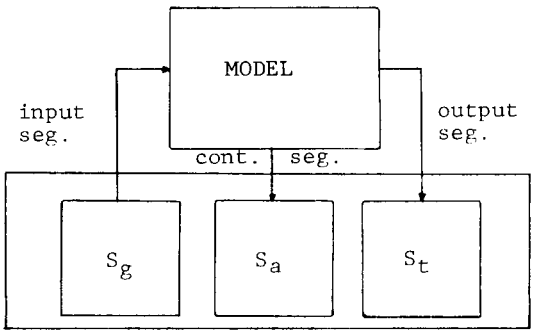

Fig. 1 Realization of an Experimental Frame

to be executed by the computer. Building on the above formalization of the experimental frame concepts, in the ensuing sections we investigate the top-down frame decomposition methodology and a mapping of hierarchically specified experimental frames onto a distributed abstract simulator.

HIERARCHICAL SPECIFICATION OF EXPERIMENTAL FRAMES

We shall base our considerations on the foundations underlying specification of DEVS systems in modular and hierarchical forms (10). Modular construction refers to the specification of a model and an interconnection of its components. Hierarchical model specification results from modular construction of component models to several levels of recursion. Such a specification is based on a composition tree in which the nodes are labelled by component systems specifications, and couplings of, and correspondences of, specifications at immediately subordinate level. This tree concept is employed to specify hierarchically constructed DEVS models. Such models are then mapped onto a hierarchical architecture of microprocessors (for details see (2,12)). To facilitate simulation studies of distributed systems it is necessary to provide an appropriate specification of the experimentation control.

Top-down Decomposition of Experimental Frames

In order to undertake a simulation study of a distributed model we have to determine experimental frames that reflect the objectives of the study and perform the experiments accordingly. Thus, an experimental frame module ought to be synthesized and coupled with the abstract simulator of a hierarchically specified distributed model as depicted

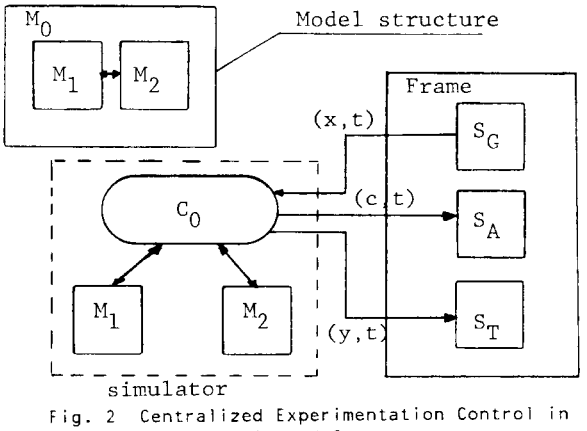

Fig. 2 Centralized Experimentation Control in Distributed System

in Figure 2.

Recall from the definition of the experimental frame realization that each component of the system S_E (i.e. generator, acceptor and transducer) is defined as a DEVS system and thus may be realized as a hierarchical coupling of systems. At this point several alternatives for experimentation control arise. In the centralized architecture as illustrated in Figure 2 the control is concentrated within the master module while the network simulators are responsible for execution of model component dynamics.

The coupling of the frame module S_E and the abstract simulator are defined as follows: the generator S_G originates the messages (x,t) that are received by the root coordinator C_0 (for description of the coordinator see (10,12)) as external events to the model. The output statistics are gathered by collecting the (y,t) message from the root coordinator. This message defines an input signal to the frame transducer S_T. It carries the information corresponding to the changes of output variables in each subordinate DEVS component. (For the sake of clarity we shall not employ a separate transducer to process the output segments and produce summary mappings. The extension that implements such a device is straightforward).

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

Such a centralized architecture may appear attractive since it involves a single experimental frame module directly linked to the global coordinator. However, the realization of the components S_G, S_A and S_T might be very complicated due the complexity of the functions that they perform. Secondly, it is our objective to define a hierarchical frame representation that can be mapped onto a distributed simulator. The components of an experimental frame should be distributed as well and their ultimate software and/or hardware realization can be constructed by appropriately linking off-the-shelf elements. In fact the realization of the experimentation in a similar context is discussed by Dekker (the concept of a cosystem) and Oren (GEST implementation of local frame segments) (1,5).

We proceed to establish the principles of top-down decomposition of experimental frames. Let us consider the aspect of input generation first. Assume that the model M has two components M_1 and M_2. In the centralized mode of experimentation a generator G has to be defined and coupled to M_0 through its input ports. Figure 3 depicts the above situation. In order to realize G as a coupling of component, possibly less complex, generators we have to identify the structure of the input segments received by the model. In the most general case we can assume that an input segment is decomposed into mutually independent segments ω_1 and ω_2 that are applied directly to model components M_1 and M_2 and the segment ω_0 which accounts for input to their coupling i.e. M_0. In other words G generates segments $\omega=(\omega_0,\omega_1,\omega_2)$. We decompose G into generator G_0, G_1 and G_2 and couple them with the model components accordingly. The hierarchical

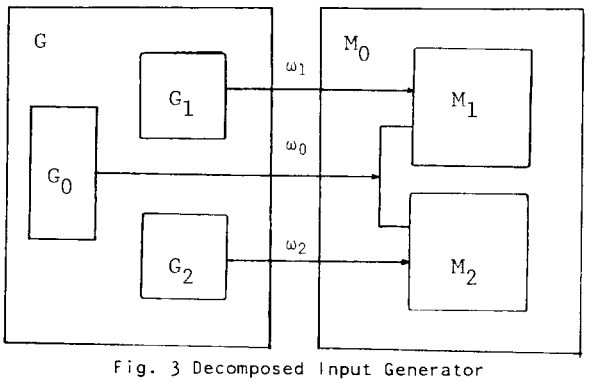

Fig. 3 Decomposed Input Generator

specification of the input generator is given by the following theorem:

Theorem 1. The parallel coupling of the DEVS generator $(G_0||G_1||G_2)$ is a hierarchical decomposition of the DEVS generator G if it simulates G.

Proof: The parallel coupling of component generators is a DEVS in a modular form (10) in which no component is an influencee or an influencer of another component. The simulation relation is defined as a homomorphism between two DEVS systems. (For detailed proof see (7)).

The physical realization of the above decomposition can be achieved by applying the DEVS projector to the global input segment ω (2).

Notice that any model component may itself be composed of submodels. Thus, the 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.

The decomposition process of the output transducer is similar to that of the input generator. The transducer T collects global output segments $\rho=(\rho_0,\rho_1,\rho_2)$ where ρ may represent correlated output of the components M_1, M_2 while ρ_1 and ρ_2 are mutually independent, local output segments. We carry out the decomposition of the output transducer as follows. T is decomposed into T_0, T_1 and T_2 that are coupled to the model components M_0, M_1 and M_2, respectively (Figure 4).

The realization of such a decomposition process can be achieved by using the output DEVS abstracter (2). The hierarchical specification of the output

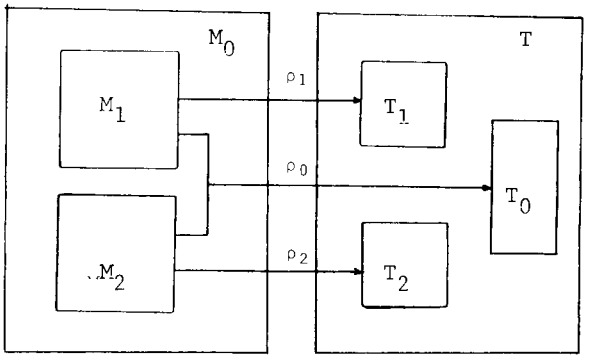

Fig. 4 Decomposed Output Transducer

transducer is defined in the manner analogous to the specification of the input generator.

Notice that the above specification establishes observation frames at any two subsequent levels of the system 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_0 is decomposed in exactly the same way as the output transducer.

The component acceptors A_0, A_1 and A_2 monitor the run control trajectories μ_0, μ_1 and μ_2, respectively. Conceptually, A_0 checks for acceptance of the global run control segment pertaining to M_0 while the components acceptors monitor the control segments local to M_1 and M_2. 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.

Having provided the framework for the top-down decomposition of input, output and run control components of an experimental frame we assert that the hierarchical specification of the experimental frame is given by the following proposition.

Proposition

Let a hierarchical DEVS structure be given by the composition tree Tree(S,C,M)=<T,m>, where S is the set of DEVS models, C={C|C=<D,{I_d}, {Z_d}>}, where C is a coupling scheme consisting of an indexing set D, an indexed family of subsets of D and an indexed family of functions; M is a set of DEVS isomorphisms (10). A coupling scheme C consists of an indexing set, an indexed family of subsets of D (the potential influencee sets), and an indexed family of functions (the potential output translation sets).

We define the associated experimental frame composition tree as the following tuple:

ExpTree(E,C,M)=<T',m'>

where T' is a finite tree, m' is a mapping, the node labelling of T' subject to constraints:

m': interior_nodes(T')->E x C x D

m': leaf_nodes(T')->E

m' assigns to each interior node (including the root) a triple consisting of a frame specification, a coupling scheme C for coupling the frame specifications at a node and the successors of the node, and the correspondence D for comparing the frame specification with the resultant of the coupling of its components. The leaves receive only the atomic frame specifications which are not further decomposable. D is a set of correspondences underlying the morphisms in M, in particular the frame derivability relations (6,10). The experimental frame composition tree is depicted in Figure 5.

The coupling scheme C is defined as a parallel composition of frame components i.e. generators, transducers and acceptors. Since the frame components are DEVS systems, the coupling can be carried out by using the DEVS specification in the modular form

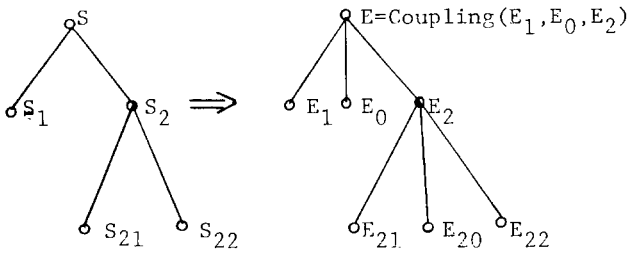

Fig. 5 Experimental Frame Composition Tree

(10). The resulting frame composition tree is structurally isomorphic with the model tree. This fact facilitates the mapping of a hierarchical frame specification onto the abstract hierarchical simulator. The ultimate purpose of such a combined mode/frame distributed architecture is to realize it on a network of microprocessors. In fact a study of this subject is under way and is presented in the companion paper (3).

MAPPING HIERARCHICAL SPECIFICATION OF EXPERIMENTAL FRAMES ONTO THE ABSTRACT SIMULATOR

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

1.) The coupling of the simulator and frame must be closed i.e. must result in an abstract simulator.

2.) 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.

3.) The overhead generated by the mapping should be minimal, i.e. the degree of parallelism achieved by the hierarchical architecture of the simulator without the frame components should not decrease below a certain level of satisfaction.

Motivated by the above specified 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 now simulate the model with a pertinent experimental frame. Recall that the frame components are defined as DEVS systems and thus can be simulated by an abstract simulator as well. However, coordination is required between the simulators of the model, generator, acceptor and transducer. To provide for such a coordination we introduce the concept of a model/frame coupler (MFC). An MFC is a coordinator (as defined in the abstract simulator (2,10,12)) which performs the following functions. At the level local to its frame and model (i.e. level i) it 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 (10). This significantly simplifies the design of the MFC since it has to schedule the internal transitions of only one active component i.e. the generator. The coupler serves also 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) and (m,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 6. (Figure 2 illustrates the model structure). The coupler MFC_1 coordinates the 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) coordinator C_0. The role of MFC's 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 in spite of the fact that C_0 is the root coordinator it is still necessary to simulate the model M_0 and its frame E_0. To achieve that an 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. It 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.

It is easy to notice that the mapping of a hierarchically specified frame onto a distributed simulator (if defined in the above terms) can be carried out recursively.

At this point we should examine the proposed coupling and determine to what extent it is consistent with the requirements that we specified at the beginning of this section.

The mapping results in an abstract simulator that correctly simulates the combined model/frame DEVS. To verify this, observe that the components of frames are simulated by the DEVS simulators and that the model/frame couplers are coordinators. The correctness of the DEVS simulator and coordinator has been proven in (10). (For detailed proof of the correctness of the mapping see (7)).

Since a means for coupling of an experimental frame to a model component at any level of the hierarchy are provided, it is apparent that the maximum decentralization of the experimentation can be achieved. We have not established yet to what extent the mapping of frames onto the abstract simulator effects the degree of parallelism provided by the simulator's architecture. It should be noted, however, that the communication between a model and its frame is restricted to lateral exchanges of messages through the MFC. Further study of the properties of the coupling is currently under way (7).

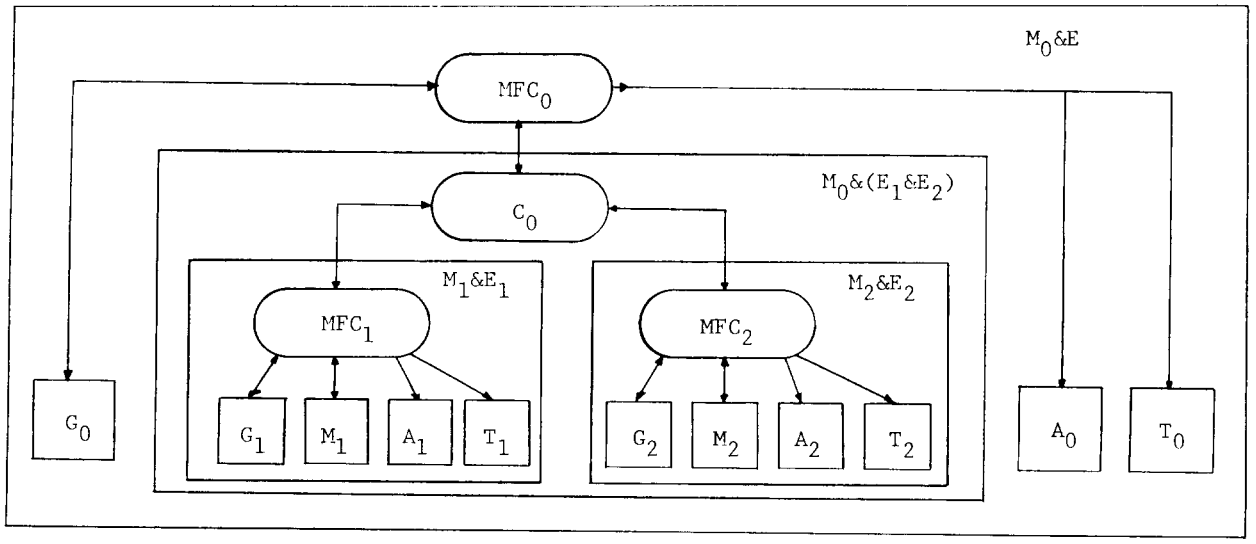

Fig. 6 Abstract Simulator of Distributed Model with Experimental Frames.

An alternative approach to realization of the frame/simulator mapping is based on the same principle of behavior. The frame components are embodied in the coordinators at each level. Namely, each coordinator is augmented with the functions performed thus far by the model/frame coupler at the level subordinate to that coordinator. Such a realization decreases the global number of coordinators used at the cost of increased complexity of each unit. The performance measures of both architectures are still being investigated (2,7,11).

EXECUTION CONTROL IN THE DISTRIBUTED ABSTRACT SIMULATOR

Recall that the third component, after model and experimental frame specifications, to a simulation program is the execution control module (10). This module is responsible for selecting the finite set of all possible simulation runs that are to be executed on the model with its frame. To start a simulation run the execution control module has to set the initial state of the model and the initial states of the frame components i.e. generators, acceptors and transducers, and initiate the execution of model/frame simulators. To terminate an experiment the module must determine whether the global termination condition holds. The final states of the acceptors monitoring the model run control trajectories are observed and the termination condition is evaluated based on the values of these states.

We propose that the execution control module for the distributed abstract simulator be realized as a unit providing a user interface, simulation data base, model data base, and simulator as illustrated in Figure 7. To initiate a simulation run the modeller should set the simulator into a desired state using the user interface. This process should be supported by the knowledge concerning the range set of variables, the minimal state variable sets and the intervariable relations constraining the values which may be simultaneously assigned to the state variables (10). The model data base should contain this type of information and ought to be accessible to the user via the interface.

During the simulation run the control module should monitor the state of acceptors and determine whether the experimentation is to be continued or terminated. The conditions for continuation or termination may be determined by the modeller when the acceptors are defined, or alternatively, the final states of the acceptors may be parametrized and are to be instantiated via the control module at the beginning of the run.

Another problem arises in case the modeller wants to continue the simulation run after it has been terminated so as to produce the same trajectory as would have been produced if the run had not been interrupted. The simulator is set in the same state that it was in at the time of termination. This can be achieved by retrieving the pertinent information about the state of the simulator from the simulation data base. In our future research we shall develop methods for interactive parameter exploration in distributed simulators based on the approach outlined by Zeigler in (10).

CONCLUSIONS

This paper has proposed a methodology for specification of experimental frames in hierarchically specified distributed discrete event systems. The approach consolidates efforts to provide a unified

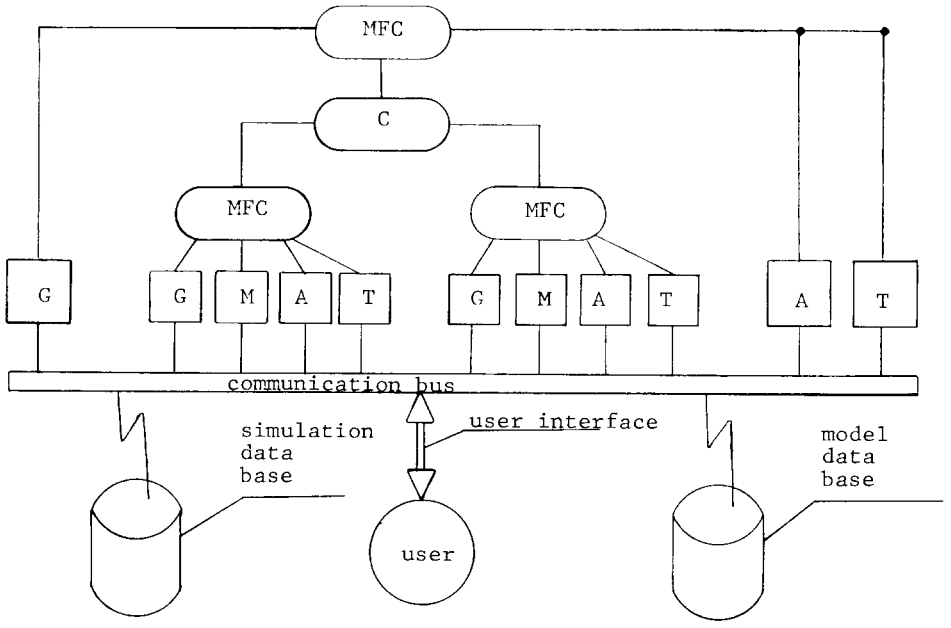

Fig. 7 Experimentation Control in Distributed Simulator

framework for simulation of distributed systems. While many of the presented concepts are still of a propositional nature, they are being further investigated. We shall seek to determine the performance measures of the frame/simulator coupling and further develop the concept of the execution control module. The implications to procedural and hardware realizations of the presented concepts will also be investigated.

ACKNOWLEDGMENTS

I wish to thank Prof. Bernard P. Zeigler for inspiration and many helpful discussions along the way.

REFERENCES

1. Dekker L., "Concepts for An Advanced Parallel Simulation Architecture", in Simulation and Model-Based Methodologies: An Integrative View, Springer-Verlag, New York 1984.

2. Concepcion A., "Distributed Simulation on Multiprocessors: Specification, Design, and Architecture", Doct. Diss. Dept. of Computer Science, Wayne State University., Detroit, Michigan, 1984.

3. Concepcion A., "Mapping distributed Simulators onto the Hierarchical Multi-Bus Multiprocessor Architecture", Proc. of Distributed Simulation 1985, San Diego, 1985.

4. Oren T.I., "Computer Aided Modelling Systems", in Progress in Modelling and Simulation, Academic Press, London, 1982.

5. Oren T.I., "GEST - A Modelling and Simulation Language Based on System Theoretic Concepts", in Simulation and Model-Based Methodologies: An Integrative View, Springer-Verlag, New York 1984

6. Rozenblit J.W., "EXP - A Software Tool for Experimental Frame Specification in Discrete Event Modelling and Simulation", in Proc. of the 1984 Summer Computer Simulation Conference, pp. 967-971, Boston 1984.

7. Rozenblit J.W., "Realization of Experimental Frames in Multi-Objective Simulation Modelling", Doct. Diss., Dept. of Computer Science, Wayne State Univ., Detroit, Michigan. (in preparation).

8. Zeigler B.P. "Structures for Model Based Simulation Systems", in Simulation and Model-Based Methodology: An Integrative View, Springer-Verlag, New York, 1984.

9. Zeigler B.P. "Modelling and Simulation Methodology: State of The Art and Promising Directions", Simulation of Systems '79, North Holland, Amsterdam, 1980 pp. 819-835.

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

11. Zeigler B.P. and Baik D.K. "Performance and Parallelism Efficiency in Hierarchical Simulators" (in preparation).

12. Zeigler B.P., "Discrete Event Formalism for Model Based Distributed Simulation", Proc. of Distributed Simulation 1985, San Diego, 1985.