Proceedings of the 1993 Winter Simulation Conference
G.W. Evans, M. Mollaghasemi, E.C. Russell, W.E. Biles (eds.)

AN ENVIRONMENT FOR AUTOMATIC SYSTEM PERFORMANCE EVALUATION

Lien-Pharn Chien
Jerzy W. Rozenblit

Department of Electrical and Computer Engineering
The University of Arizona,
Tucson, Arizona 85721, U.S.A.

ABSTRACT

A wide range of modeling and simulation packages have been applied to evaluate manufacturing systems, computer systems, and telecommunication networks. The objective of using simulation is to assess system designs prior to their implementation. In this paper, a modeling and simulation environment supported by a hierarchical, model-based management methodology is presented. This environment allows users to efficiently construct system models and procedures for performance evaluation without requiring prior knowledge of the simulation description language. The environment, called Performance index-Oriented modeling and Simulation Environment (POSE), provides hierarchical function manipulation based on the partition of the system's architecture. A window-based graphical front-end has been developed to offer a simple and straightforward user interface. Through POSE, the model and simulation development time has been significant reduced. An example of a manufacturing system is presented to illustrate POSE.

1 INTRODUCTION

Due to complex, multilayer, multicomponent designs of new systems, it is critical to provide an efficient means for performance data generation. In this paper, we present an environment called Performance index-Oriented modeling and Simulation Environment (POSE) to automate system modeling and system simulation. POSE uses DEVS (Discrete EVent System Specification) (Zeigler 1990) as an underlying simulation engine. Our environment features the following:

1. Automation of model development.
2. Hiding of simulation description language from the user.
3. Efficient, simulation-based performance evaluation.

To structure the model development process, POSE uses the composition tree and system entity structure concepts (Zeigler 1984, Rozenblit and Zeigler 1988). To manage performance evaluation, we use the experimental frame formalism (Zeigler 1984).

In the ensuing sections, model design and development using functions provided by POSE are addressed. Each function is discussed in detail.

2 MODEL DESIGN USING POSE

Our environment is intended to automate the model creation for use in simulation and to provide related performance data calculation and system evaluation. We first consider the requirements and constraints of the system to be modelled. After considering the system's requirements, we then proceed with the modeling procedure. We develop two types of models: a) the Experimental Frames (EFs) used for generating and processing performance statistics, and b) models needed to simulate the proposed or the real system in question. The model development procedure is one of the major tasks performed by POSE. It is organized hierarchically. The hierarchical model construction shown in Figure 1 includes four steps: 1) Automatic Generation of Experimental Frame (AGEF), 2) Node Modeling, 3) System Modeling, and 4) Model Integration (MI). The Node Modeling and the System Modeling are combined and are termed Automatic Generation of System Model (AGSM). The goals and functions of AGEF, AGSM and MI are explained below:

1. *Automatic Generation of Experimental Frame*: This function is to automatically generate the experimental frames (EFs). The elements required to generate EFs are the workload distribution of the system being modelled, the requirements of

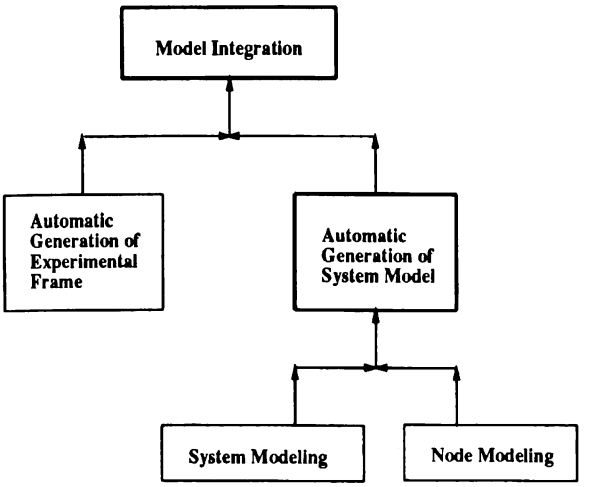

Figure 1: Hierarchical Model Construction in POSE

the system, and the object-based library called the Generic Experimental Frame Base (GEFB) (Rozenblit 1991). All the created EFs are stored in the Experimental Frame Model Base (EFMB).

2. *Automatic Generation of System Model*: The elementary atomic models (nodes) in the system being modelled are constructed during the Node Modeling procedure. Then, an overall system model (network) is established through coupling the elementary models and/or other networks (as subsystems). Both node and network models are stored in the System Model Base (SMB), and can be invoked during model integration (MI).

3. *Model Integration*: Models in the EFMB or the SMB can be processed in the DEVS-Scheme environment (Zeigler 1990) in the standalone mode. Through a proper coupling of models from the model base, at the integration stage, a complete coupled model is created. After simulation runs are completed, related performance statistics are evaluated. The integrated models are saved in the Integrated Model Base (IMB) for future reuse.

Along with the operations of the AGEF, AGSM, and MI, three model bases (EFMB, SMB and IMB) and an object-based library (GEFB) are required. These bases are initially empty. They are populated when models are developed in POSE. The efficiency and performance of the POSE package is greatly enhanced by the hierarchical organization of these bases. This feature can be utilized in different ways. The models existing in SMB can be accessed and incorporated into a larger and more complex system model without any restrictions as long as the behavior of the new system model is captured and data are processed properly. This construction results in the system model being set up in a hierarchical manner. The same process takes place during the MI execution in which EFs in the EFMB and system models in the SMB are retrieved.

In the GEFB, the performance objects in the library are related to one another. This close relationship results in a hierarchical performance chain which we explain in the next section.

Due to its hierarchical structure, POSE facilitates flexible and rapid modeling. Suppose that an integrated model with incorrect performance data is found. The system designer (POSE's user) identifies the incorrect subsystem model. There are two possible solutions for this problem in POSE. One way is to couple a correct subsystem model through the AGSM (if the subsystem model exists in SMB). A correct subsystem could also be created from scratch by using the AGSM. Once the subsystem has been corrected, the designer can reuse the existing integrated model, bypass the step of model integration, and perform model simulation.

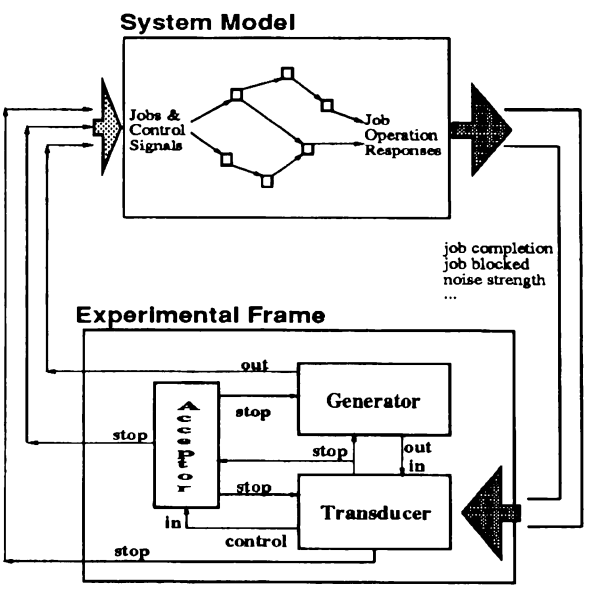

Figure 2: The Relationship Between an EF and The System Model

In the next section, we describe the Automatic Generation of Experimental Frame. As we have stated previously, experimental frames are a means of generating performance related data. The structure of an EF and its relation to a simulation model is shown in Figure 2.

3 AUTOMATIC GENERATION OF EXPERIMENTAL FRAME - AGEF

An experimental frame (Zeigler 1984) can be defined as a coupling of a generator, a transducer and an acceptor. Various EF configurations are designed to reflect the diverse requirements and conditions associated with specific applications models. These configurations are organized through a window-driven procedure developed for the AGEF as shown in Figure 3. Since POSE is targeted for performance evaluation, special features have been incorporated in an EF structure. These features are discussed below.

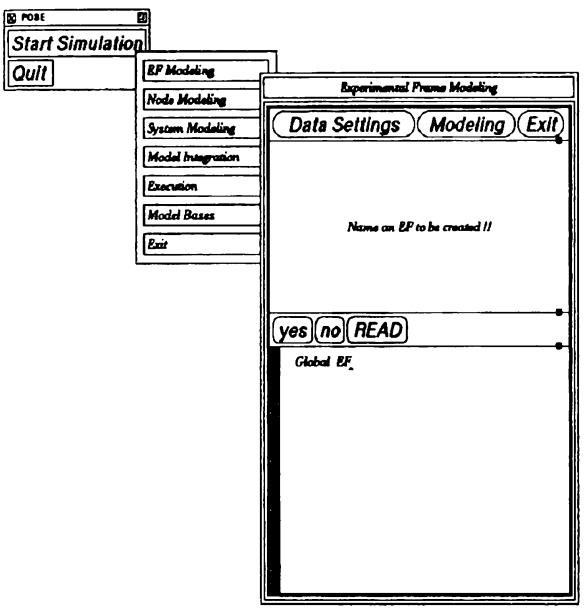

Figure 3: The Window-Driven AGEF

Generators Characterized by Probability Distributions: In order to emulate a real system's behavior, a realistically driven workload is an important factor. The common discrete and continuous probability distribution functions are incorporated into the generator (Law and Kelton 1991). To ensure the correctness of implementing the discrete and continuous distribution functions is nontrivial. Several approaches have been introduced in (Law and Kelton 1991). The *Inverse Transform* approach is adopted here and has been validated through the method of "histogram", the expected value, and the variance.

Transducers Used for Performance Calculation: A transducer should collect raw performance data and process them in terms of the pre-defined performance metrics during a simulation session. These raw and processed performance data are stored in the corresponding log files for analysis and evaluation. Furthermore, by considering the random behavior of the system's workload and processing rates, a transducer needs to decide if a system's state is in equilibrium during the simulation.

Acceptors As System Runtime Controllers: The major feature provided by an acceptor is the ability to define a set of system constraints used for runtime control. Therefore, an acceptor can periodically check whether the constraint boundary is broken or not. If a violation occurs, an immediate warning signal is sent out to the model(s) coupled to it.

3.1 Generic Experimental Frame

In order to achieve high efficiency and flexibility in the construction of the EF models, the concept of Generic Experimental Frame (Rozenblit 1991), and the scheme of Performance Metric Tree (Rozenblit and Hu 1989) are applied.

A generic experimental frame (GEF) is an universal performance metric specification from which diverse sorts of experimental frames used for performance evaluation in different fields can be derived. To illustrate the generic frame concept consider the following case: in an automated manufacturing system, performance metrics could deal with the throughput of a specific tooling machine, the mean turnaround time per product at a workstation or the whole system. A GEF consists of a set of variables that correspond to each of the performance metrics. The construction of the GEFB is primarily based on the concept of GEF and the structured characteristic provided by a frame representation (Rich 1983). Every entity in the GEFB is called a *performance object frame* (*POF*). It contains information about a performance object's metric (an algebraic expression) in a frame structure (Rich 1983). The *POF*s in the GEFB need to be systematically chained together for the sake of efficiency and flexibility of the EF modeling procedure. This systematic management is carried out by utilizing the scheme of Performance Metric Tree (PMT).

A PMT is a data structure built upon an algebraic relation of the performance metric under which the system is to be evaluated. For example, consider Figure 4 which shows a PMT for evaluating a machine's utilization. The root node of the tree expresses the performance measure of interest. The internal node, *Throughput* is a subexpression which also is a PMT. The leaf nodes in the tree are *atomic expressions*, i.e., specifications that cannot be further decomposed. (Note that the blocks drawn at the leaves denote *generic modules*; others represent parameters.) The generic modules are defined in the Experimental Frame Model Base (EFMB) to facili-

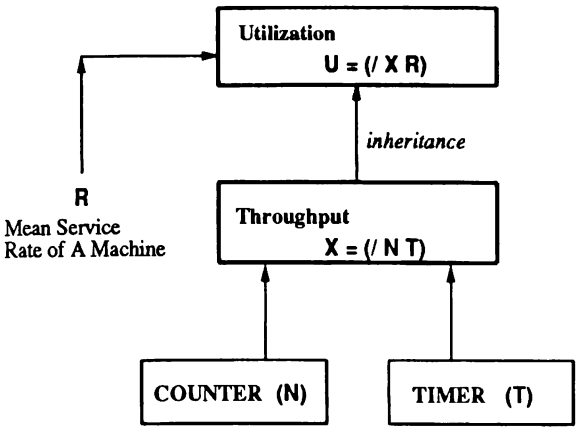

Figure 4: The Utilization PMT of The Machine

tate the translation of new performance metrics. The parameters are used to describe some known properties about the system or the component being monitored. The *Utilization* PMT inherits all the properties from the *Throughput* PMT. Therefore, the *inheritance chain* is established. Through this chain, the *Utilization* performance metric of the machine can be expressed by $U = (/XR)$. This utilization expression is based on Little's result (Little 1961). Through the inheritance chains, the *POF*s in the GEFB are organized hierarchically. This hierarchical, object-based library enables POSE's users to easily and efficiently proceed EF modeling. The EF models are generated automatically via the AGEF. They are stored in the EFMB for reuse in the Model Integration procedure.

4 AUTOMATIC GENERATION OF SYSTEM MODEL – AGSM

The AGSM procedure is divided into two phases: the node (component) modeling and the coupled system (network) modeling. A system model could have a very simple configuration, composed of only one node model, or have a complex, multilayer, multicomponent configuration. This kind of complex system model implies a hierarchical composition which is the major concern at the system modeling stage.

4.1 Node Modeling

By using the DEVS formalism (Zeigler 1984), the properties of a queueing model (Schwartz 1987) are embedded in a node. Hence, the related probability distribution functions mentioned in the construction of a generator are adopted and are used as the node processing-time (service-time) distribution functions. Node models generated through the Node Modeling procedure are stored in the SMB. They can take the following forms:

1. *Normal Node*: This node model processes an input job (an event in DEVS) within a certain time, and passes it out successfully. The processing time can be state independent or state dependent.
2. *Blockader*: It acts as an obstacle to absorb incoming jobs without producing any output. For example, a component failure beyond repair can be modelled by this type of node.
3. *Dispatcher*: A node model as a job dispatching center is used to dispatch jobs to other nodes with a user-defined probabilistic behavior. It takes zero processing time to dispatch jobs.
4. *Noisemaker*: This node model can generate errors such as different transmission medium noises in networks.

4.2 System Modeling

After the required, individual node models are created, a system model then can be constructed by coupling them. This is accomplished by an algorithm called **SM-Algo**. In order to describe this algorithm, we define the term *Closed System*. A *Closed System* is a system model obeying two conditions: 1) there is more than one model inside it, and 2) no identical subsystems excluding the *marked non-Closed Systems* can be extracted from it. The subsystem mentioned in this definition has a strict constraint, i.e., it must consist of more than one model. A method called *Make Instances* is used to make several instances from the given model. Based on these definitions, the **SM-Algo** is specified as follows:

Step 1: Analyze the system to find two sets: *Closed System* set (CS-set) and *non-Closed System* set (NCS-set). (At least one of these sets must not be empty.) Count the number of instances for every element in both sets.

Step 2: If CS-set is empty, then a complete system model can be constructed by invoking the *Make Instances* method for the element(s) in NCS-set if necessary, based on the number computed in Step 1. Delete the element(s) from NCS-set. Add the new instances to NCS-set. Set up the appropriate coupling for the element(s) in NCS-set if needed. Terminate the algorithm. Otherwise, go to Step 3.

Step 3: Set up the proper coupling for each element in CS-set.

Step 4: Perform *Make Instances* on the element(s) in CS-set if necessary, based on the number computed

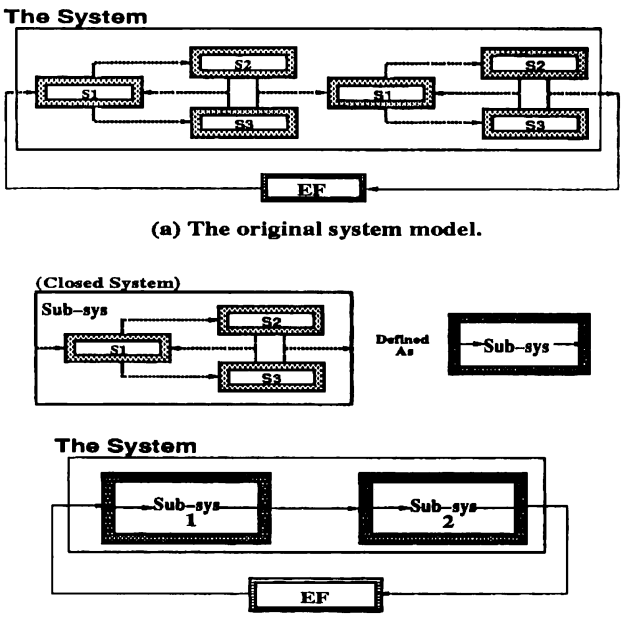

Figure 5: The Effect of The SM-Algo Approach

in Step 1. Delete the element(s) from CS-set. Add the new instances to CS-set.

Step 5: Change the property of every element in CS-set by switching it to a *non-Closed System*. Go to Step 1.

By utilizing this algorithm, we can design a complex, multilayer, multicomponent system model efficiently. As shown in Figure 5, the model (a) is the original design. The model (b) is the result of applying the **SM-Algo** procedure. The steps to convert model (a) to model (b) are:

1. After executing Step 1, we find that CS-set contains Sub-sys with two instances. NCS-set is empty.

2. Since CS-set is not empty, Step 3 is performed. Then the proper coupling is set up for Sub-sys.

3. Two new instances named Sub-sys1 and Sub-sys2 are created by invoking *Make Instances* at Step 4. Delete Sub-sys from CS-set. Add Sub-sys1 and Sub-sys2 to CS-set.

4. At Step 5, Change the property of Sub-sys1 and Sub-sys2 from *Closed System* to *non-Closed System*. Go to Step 1.

5. After analyzing the system again, CS-set becomes empty. NCS-set includes Sub-sys1 and Sub-sys2. Each element has only one instance.

6. At Step 2, the new system model (b) is constructed by coupling the elements from NCS-set.

5 MODEL INTEGRATION AND SIMULATION

The purpose of the Model Integration block is to set up an integrated model which can be executed in DEVS-Scheme. The schemes of Distributed and Global Experimental Frames (Rozenblit 1991) are adopted to attach experimental frames to the system model. The expected integrated models can be established based on the algorithm called **MI-Algo**. Two methods are used to support the algorithm. One is the *Make Instances* defined in the **SM-Algo** approach. The other is called *String Matching and Creation* (SMC). It is used for the EF only. The **MI-Algo** procedure is given below:

Step 1: Retrieve the EF(s) required from the EFMB and the system model(s) from the SMB. Save these models in the sets, EF-set and System-set, respectively.

Step 2: Perform *Make Instances* on the element(s) of System-set if necessary. Delete the element(s) from System-set. Add the new instances to System-set.

Step 3: Perform the *Make Instances* and SMC methods on the element(s) of EF-set if necessary. Delete the element(s) from EF-set. Add the new instances into into EF-set.

Step 4: Set up the coupling for the elements in the sets EF-set and System-set. Terminate the procedure.

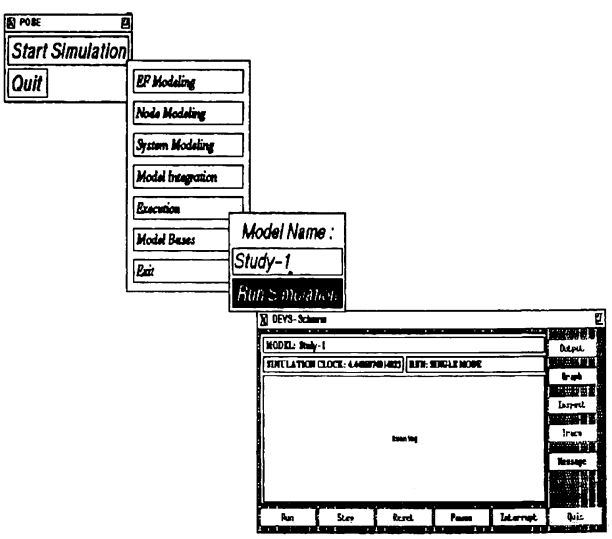

Figure 6: The DEVS-Scheme Simulation in POSE

Through the **MI-Algo** approach, integrated models are generated and stored in the IMB for simulation. The simulation stage invokes DEVS-Scheme to execute the integrated model. Figure 6 shows the DEVS-Scheme simulation window running in POSE.

6 AN APPLICATION EXAMPLE

A case study by using POSE to model and simulate a manufacturing cell (MC) (Enrick 1985, Merabet 1986) is briefly summarized. Through experimenting with various repair probabilities associated with limited and unlimited workstation's capacities, the related performance objects are analyzed.

The MC model shown in Figure 7 is composed of four machining workstations (M_1, M_2, M_3 and M_4), one repair station (RS) and a raw material dispatching station (Dispatcher). The assumptions specified in the model are:

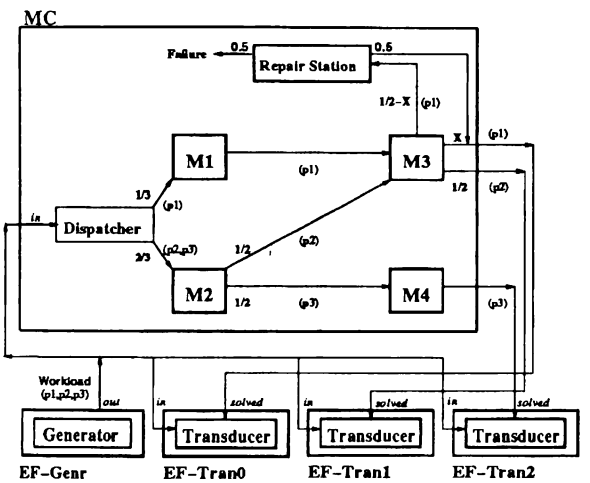

Figure 7: A Manufacturing Cell

1. The raw material is classified into three kinds: p_1, p_2 and p_3. These three kinds are generated with equal quantities by the generator.

2. The probability distribution of the generator is exponential with the mean 20 minutes, i.e., the input rate of the MC is 1 part per 20 minutes.

3. All part delivery between stations is handled by conveyors. For simplicity of simulation, the delivery time is included in the machining time of the destination station. Meanwhile, the length of each conveyor segment is assigned to be the capacity of the destination station.

4. The dispatcher is used to dispatch p_1 parts to M_1, and p_2 and p_3 parts to M_2.

5. All machining workstations have uniform probability distributions with low and high bounds (minutes): M_1 (3,5), M_2 (6,8), M_3 (10,12), M_4 (3,5).

6. The probability distribution of the RS is set to exponential with mean 20 minutes by considering the uncertainty of repairing a part.

7. Only p_1 parts are considered to be made with defect during the machining procedure in the M_3.

Two major tests are processed in POSE. One is to set limited capacities to the stations. Another is to give each station's capacity without limitation for the sake of comparison. The performance objects under consideration are throughput and turnaround. Also, this experiment is concentrated on the performance of machining p_1 parts and examining the MC's throughput. The results are summarized below (also see Figure 8): (The p_1 repair probability occurred at the M_3 is changed from 0.0 to 0.25.)

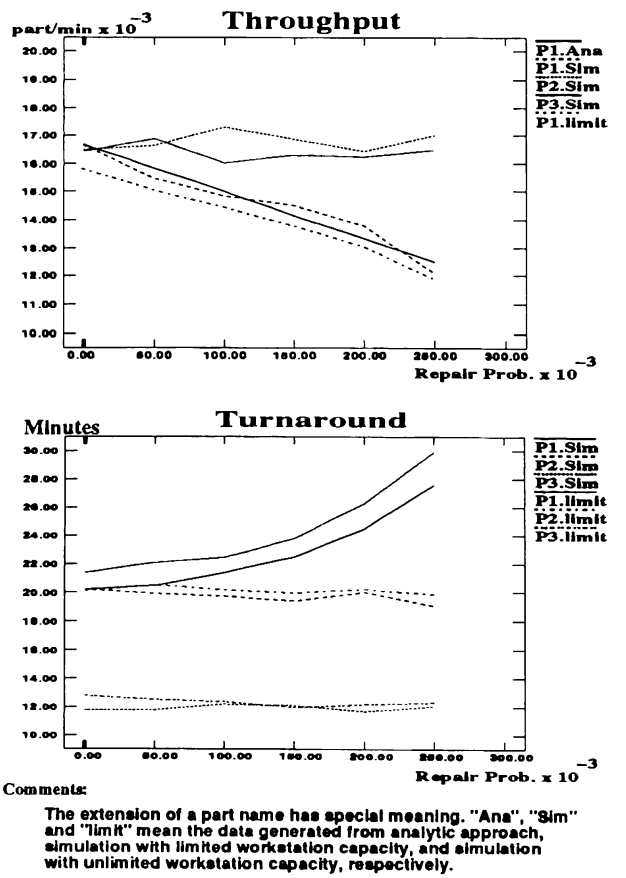

Figure 8: The Performance Outcome of The Case Study

- The throughput error percentage between the analytic approach "P1.Ana" and the simulation approach "P1.Sim", under the unlimited case, is less than 4%. The throughput of "P2.Sim" and "P3.Sim" is close to 0.01667 part per minute which matches the assumptions 1 and 2. Therefore, the simulation experiment is highly reliable.

- The throughput degradation caused by the limited capacity constraint is shown in the figure. This constraint results in 10% throughput degradation of p_1 parts, and 1% degradation of the MC's throughput. This situation becomes more serious as smaller capacity is assigned to the stations.

- As to the turnaround graph, the p_1's turnaround time grows faster as the repair probability increases. This is because more repair time is needed at the RS.

- The turnaround times of all parts are different for the limited and unlimited cases. This situation shows the blocking problem in this MC.

7 SUMMARY

We have presented an automatic modeling and simulation environment called POSE. By utilizing the modeling functions provided by POSE, users can create system models for an application environment and for performance data collection and calculation. The efficiency of the model generation procedure is attributed to the hierarchical design of the modeling functions, the model bases (EFMB, SMB and IMB), and the object-based library GEFB. Through POSE, rapid simulation and performance evaluation is achieved.

Our current work focuses on the application of Multiple Criteria Decision Making approaches to facilitate tradeoff among alternative system models.

REFERENCES

Enrick, N. L. 1985. *Quality, Reliability, and Process Improvement.* 8th ed. Industrial Press Inc., New York.

Law A. M. and W. D. Kelton. 1991. *Simulation Modeling and Analysis.* 2nd ed. McGraw-Hill, Inc.

Little, J. D. C. 1961. A Proof of The Queueing Formula $L = \lambda W$. *Operations Research* 9:383-387.

Merabet, A. A. 1986. Dynamic Job shop Scheduling: An Operating System Based Design. In *Flexible Manufacturing Systems: Methods and Studies*, ed. Kusiak, A. Elsevier Science Publishers B.V. 257-270.

Rich E. 1983. *Artificial Intelligence.* McGraw-Hill, Inc.

Rozenblit, J. W. 1991. Experimental Frame Specification Methodology for Hierarchical Simulation Modeling. In *International Journal of General Systems* 19:3:317-336.

Rozenblit, J. W. and B. P. Zeigler. 1988. Design and Modeling Concepts. In *International Encyclopedia of Robotics, Applications and Automation*, ed. Dorf, R. John Wiley and Sons. 308-322.

Rozenblit, J. W. and J. Hu. 1989. Experimental Frame Generation in a Knowledge-Based System Design and Simulation Environment. In *Modeling and Simulation Methodology: Knowledge Systems' Paradigms*, eds. Elzas, M. et al. North Holland, 451-466.

Schwartz, M. 1987. *Telecommunication Networks : Protocols, Modelling and Analysis.* Addison-Wesley.

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

Zeigler, B. P. 1990. *Object-Oriented Simulation with Hierarchical, Modular Models - Intelligent Agents and Endomorphic Systems.* Academic Press.

AUTHOR BIOGRAPHIES

JERZY W. ROZENBLIT is an Associate Professor of Electrical and Computer Engineering at The University of Arizona. His research interests lie in the application of artificial intelligence and simulation modeling to systems design. He is a member of IEEE, ACM, and SCS.

LIEN-PHARN CHIEN is a Ph.D. candidate in the Department of Electrical and Computer Engineering at The University of Arizona. His research interests include automation of modeling and simulation, design of warehouse automation systems, and knowledge-based systems associated with fuzzy theory. He is a member of IEEE.