An Approach to Object-Oriented Modeling and Performance Evaluation

Lien-Pharn Chien

Department of Information Engineering
Kaohsiung Polytechnic Institute
Kaohsiung, Taiwan

Jerzy W. Rozenblit

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

Abstract

Within the past few decades, diverse modeling and simulation tools have been applied in extensive applications. The approaches used range from programming with a specific simulation description language to automation using an icon-driven user interface. The advantage in utilizing simulation is to assess the system's performance prior to an actual implementation. Functionality, maintainability, and expansibility are the primary criteria used to make a choice of a specific tool. To strengthen these criteria, a general-purpose environment called Performance Object-oriented modeling and Simulation Environment (POSE) has been developed. The objectives of POSE are to automatically construct simulation models for the systems to be designed, to efficiently define the system performance measures, and to accurately generate the performance data expected. The environment is briefly summarized and an application study for a multiprocessor computer system is presented.

1 Introduction

As fiber optics, ultra large-scale integrated circuits, asynchronous transfer mode, and more advanced technologies are introduced, new application systems have become much more complex. The behavior of the systems is usually of high complexity and is difficult to evaluate by *analytical approaches*. It is believed that if no analytical approaches can be applied, constructing a new, complex system can be expensive, time consuming, and risky. Therefore, *simulation* or *hybrid approaches* are explored to mitigate the problems [7]. This is the first reason triggering this work. In addition, it is necessary to construct a model required before carrying out system simulation. To build the system model in most of existing simulation languages, users (or system designers) must know the syntax of a specific language and how to program the model correctly. The situation motivates our research to devise a way that allows users to do system modeling without the knowledge of an underlying simulation language.

Furthermore, the object-oriented (for short, OO) concept has shown a great potential in extensive applications, especially the advantages of reuse and maintainability. The third characteristic in *POSE* is to utilize the OO concept in cooperation with Queueing Theory [4] and the structure of DEVS formalism [13] such that each *POSE*'s model has a concrete configuration and is efficient in processing the problems of system performance measures. Lastly, we strive to improve *POSE* to strengthen its functionality and expansibility. This is achieved through the design of the hierarchical model-base management. The hierarchical model bases are established by dividing the modeling procedure into two parts: the system-architecture and the system-performance modeling stages.

2 The Design of POSE

As described in detail in [1], the design concept focuses on automating the simulation model creation and providing the required performance calculation and evaluation. *POSE's* architecture is depicted in Figure 1. The arrows in the figure show the operation flow in *POSE*. Each item beside an asterisk expresses the basic part corresponding to the stage right above or below it. According to the flow, the requirements and constraints of the system to be developed are considered first. The requirements include performance objects (indices) like throughput, utilization and turnaround time. After analyzing the system's requirements and constraints, the AGEF (Automatic

0-8186-6440-1/94 $04.00 © 1994 IEEE

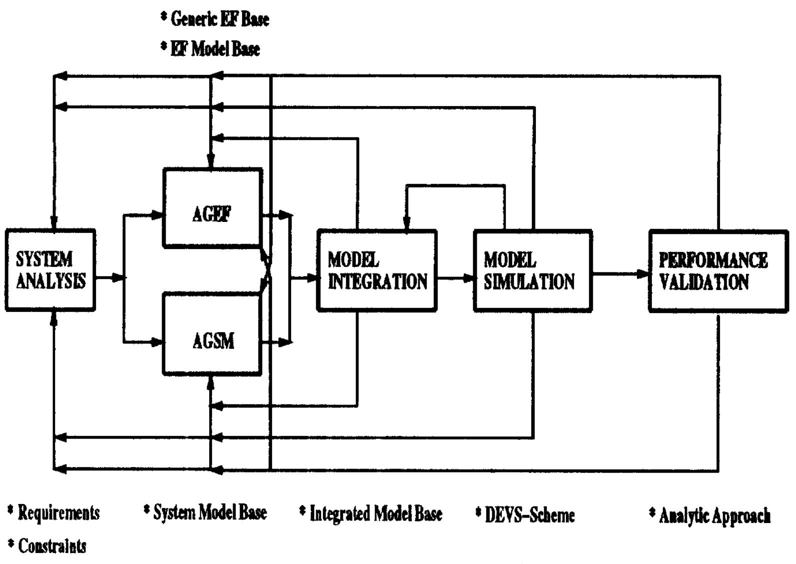


Figure 1: The Design Flow of POSE.

Generation of Experimental Frame) stage is processed in order to embed this information into an experimental frame (for short, EF) for future use [12].

At the System Analysis stage, the system's architecture is analyzed. For example, when a multiprocessor computer system is to be modelled, the information about the connections among CPUs, memory units and IO units, the characteristics of each unit, and the partitions of the system, have to be obtained after the analysis procedure. Based on this information, the AGSM (Automatic Generation of System Model) is invoked to model the computer system. As soon as the AGEF and the AGSM stages are completed, the Model Integration (for short, MI) stage takes place. A complete integrated model is then generated. This model is able to produce the performance data for the system in terms of the requirements and constraints specified in the EF(s).

The output provided by *POSE* are integrated models which are useful at the next stage, Model Simulation. All performance data are collected and computed within this stage. These data are used to validate the accuracy of the system models (e.g. the computer model) via mathematical approaches.

Three model bases, Experimental Frame Model Base (EFMB), System Model Base (SMB), and Integrated Model Base (IMB), along with a performance object-based library called Generic Experimental Frame Base (GEFB), are used to support the hierarchical modeling-automation flow. These bases have the hierarchical relationship of IMB at the root with two children SMB and EFMB. In turn, EFMB requires the resource in GEFB. They are originally empty but become populated as systems are developed in *POSE*. The power of *POSE* is enhanced by maximizing the flexibility of the execution flow feedback as referred to in Figure 1 and designing a corresponding interface shown in Figure 2 (where both *Node Modeling and System Modeling* functions comprise the AGSM stage). More details about the environment and its implementation can be found in [1]. In what follows, we focus on the model integration and simulation stage and provide an illustrative example.

3 Model Integration and Simulation

The models in the system model base (SMB) and the experimental frame model base (EFMB) can be operated in a stand alone mode with DEVS-Scheme but no meaningful output is produced. To come up with the performance metrics required for an application system, the function of Model Integration (MI) is then designed. Figure 3 shows the relationship among the SMB, EFMB and the integrated model base (IMB). This figure also points out that only integrated models are allowed to be simulated in *POSE*.

In order to carry out flexible model integration,

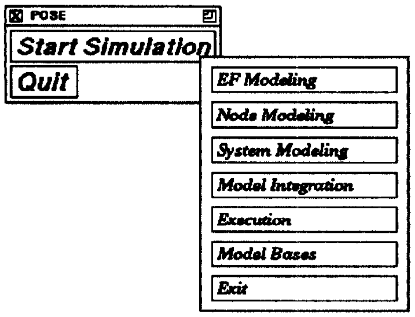


Figure 2: The Function Layout Window in POSE.

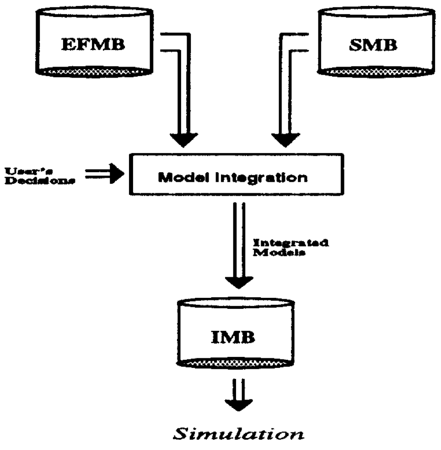


Figure 3: Model-Base Relationship in Model Integration.

the schemes of *global* and *distributed experimental frames* [8] are employed at this stage. For instance, an interconnected network like ARPANET [11] connects many Local Area Networks (LANs) via gateways. Since the role of gateways within the interconnected network is very critical and sensitive, performance measures at gateways are particularly important. In general, throughput and utilization are factors of greatest concern. This situation requires attaching different configurations of EFs to the gateways at different geographical areas. This flexibility has been carried out by using both schemes. Due to the flexible attachment of an EF to any layer or component in a system model, the EFs stored in the EFMB can be retrieved and coupled to the system model without any restriction during the processing of the MI function.

The goal of simulation in *POSE* is to generate performance data for performance evaluation. All the data expected are gathered and saved in different log files specified in the transducer(s) during simulation. Basically, there are three pre-defined log files: *job arrival*, *job finished*, and *summary*, at each transducer. Both *job arrival* and *finished* files keep the so-called raw performance data consisting of each job name and its priority with the arrival or departure time, respectively. The log files rather than the pre-defined ones are for specific purposes such as throughput, turnaround, etc. The *summary* file periodically collects all processed performance data such as:

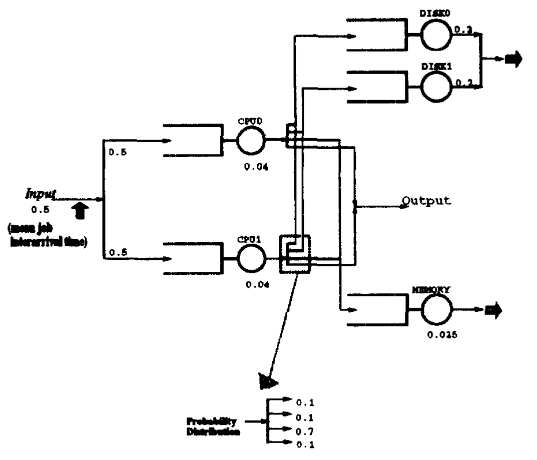
```
** ef-computer at time 320

throughput :: 2.08524590163934
turnaround :: 1.66049869504983

** ef-computer at time 340

throughput :: 2.08709677419355
turnaround :: 1.65773744063163

** ef-computer at time 360

throughput :: 2.0952380952381
turnaround :: 1.6632224979867
```

A simulation run is finished as soon as the tolerance condition pre-set by employing the technique of *terminating simulation* is met [5].

4 The Experiments in POSE

Even though *analytical approaches* provide efficient and accurate ways to process performance measures, the *simulation approach* offers an alternative when: 1) the complexity of a system prohibits deterministic results, or 2) the complexity is difficult to analyze mathematically. The *Simulation approach* also provides a means for evaluating and comparing new systems prior to their actual implementation. Nevertheless, *analytical approaches* can be used to evaluate the performance outcome generated by *POSE*.

The following simulation experiments are used to test *POSE*'s functionality and accuracy. Their simulation results are synthesized through the scheme of *confidence interval* under the control of *terminating simulation* [5]. Also, the results are evaluated by means of mathematical analysis with queueing theory.



Comment : The values beside circles are their mean job service time (ms).

Figure 4: A Multiprocessor Computer System.

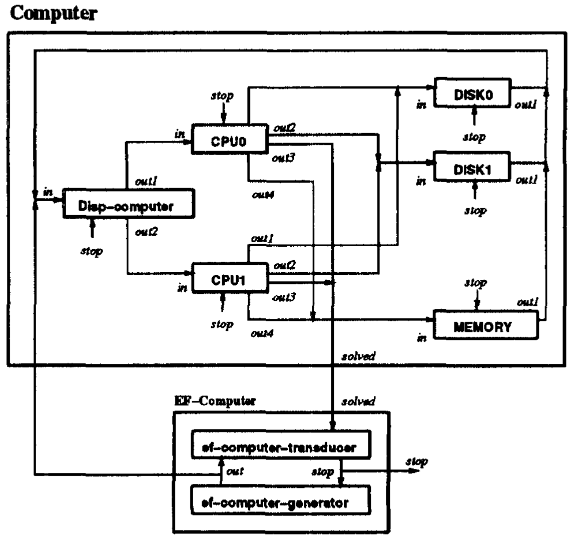


Figure 5: A Multiprocessor Computer System.

A Multiprocessor Computer System: Performance evaluation in various computer systems has been studied extensively [3, 6, 7, 10]. For comparison purposes, a multiprocessor computer system is designed in *POSE*. The experiments related to the proposed computer system shown in Figure 4 are to determine how the system would perform under various changes. Based on the figure, these changes could be about the input rate (system workload), the service rates of cpus, (main) memory and disks, and the probability settings on links (buses). Since the role of the cpu and memory is more sensitive to the whole system, we are primarily concerned with changes in their rates. Due to the variety, the performance measures regarding the mean job turnaround time, i.e., average time delay, in the system are considered. These performance measures are gathered through executing the system's model, which is created by *POSE*, as shown in Figure 5.

The first experiment examines the quantity changes in turnaround time by gradually modifying the cpu's mean service time (the reciprocal of service rate). Other factors involved here have the following conditions:

1) No jobs are blocked at any receiving unit.
2) Mean job interarrival time to the system :
 0.5 *millisecond/job*
3) Mean service rate of the memory :
 0.025 *millisecond/job*
4) Mean service rate for each disk :
 0.2 *millisecond/job*
5) Input jobs are immediately dispatched to each cpu with equal probability.
6) Cpu's outgoing jobs sent to the two disks, the memory, and outside of the system with the probabilities, 0.1, 0.1, 0.7, and 0.1, respectively.

By means of multiple simulation runs at each cpu setting, Figure 6 plots the related confidence interval with 95% via two curves **Simulation-Upper** (i.e. the upper bound of the interval) and **Simulation-Lower** (i.e. the lower bound). This area between the upper and lower curves shows that *POSE* provided a good enough estimate by comparing it to results calculated with the *analytical approach*. Based on the same conditions, we proceed with the *analytical approach* by using queueing theory. Since the computer system model is an *Open Queueing Network* (OQN) with Poisson input rate, exponential service rates and infinite buffer sizes at all queues, it can be analyzed by applying Jackson's theorem [2]. The corresponding analytical curve is marked with **Analytical-0.5** for the purpose of evaluation. (The "0.5" expresses the mean job interarrival time to the system is 0.5 millisecond, i.e., the job input rate is 2 *jobs/ millisecond*.) From the curve distributions, it is concluded that:

1) The area specified by the 95% confidence interval almost covers the analytical curve except for the range close to 0.08 and up. This exceptional range results from job congestion occurred in cpus to the extent that

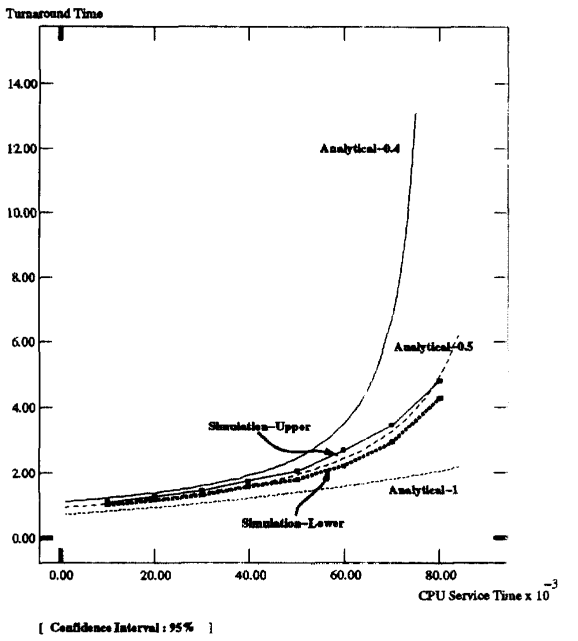


Figure 6: The Effect of The CPU's Rate Changes.

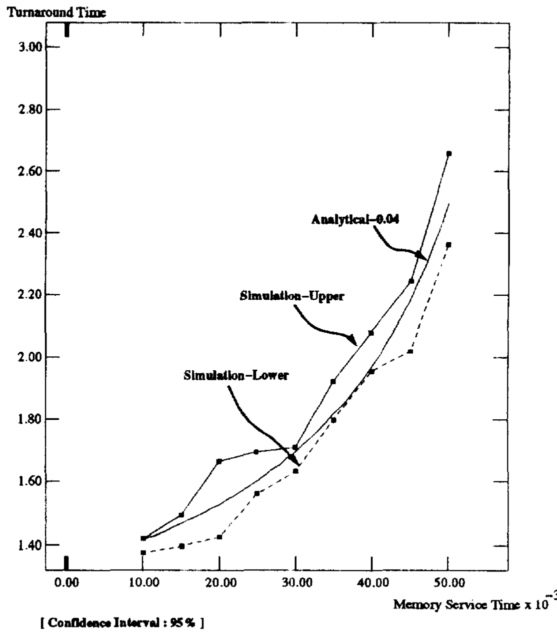


Figure 7: The Effect of The Memory's Rate Changes.

the whole system becomes unstable and the bound of 95% coverage is no longer obeyed. Therefore, this evaluation shows a high accuracy and sensitivity for the ***simulation approach*** performed in *POSE*. (In the figure, two other analytical curves are drawn for reference. The **Analytical-0.4** curve exhibits serious job congestion when the mean service time of a cpu is over 0.06, a situation that does not occur in the **Analytical-1** curve during the changes of cpu time. This is because a lower input rate is assigned to the latter.)
2) If the service time of a cpu is less than 0.06, then higher input rates are suggested unless there is a real-time factor.

The second experiment investigates how a change in memory service-time affects job turnaround time in the proposed system. Figure 7 illustrates the effect caused by the change. The related conditions set in the experiment are the same as in the first experiment except that : a) the cpu service-time is fixed at 0.04 ***millisecond/job***, and b) the memory service-time is adjusted from 0.01 to 0.05 ***millisecond/job***.

The simulation outcome is plotted by two curves **Simulation-Upper** and **Simulation-Lower** with 95% confidence interval. The related mathematical method is also built according to Jackson's theorem. The corresponding analytical outcome is drawn for comparison to the simulation outcome. Due to the nonexistence of the job congestion problem given the testing conditions, the **Analytical-0.04** curve is completely covered within the area between the two bound curves.

The experimental results for the proposed computer system show that performance outcomes produced by *POSE* provide accurate estimates. The same outcomes of the tests of Gordon-Newell Networks are also obtained.

5 Conclusions

In *POSE*, users can systematically construct a complex system model with a multilayer and multicomponent architecture through the interactive window-driven interface. The architecture facilitates hierarchical modeling and a hierarchical model-based management. By making connections from *POSE* to DEVS-Scheme, model simulation and performance data collection and computation are then accomplished. In conclusion, the contributions of this work to the field of modeling and simulation automation are in a) hiding of simulation language, b) modeling automation, c) hierarchical modeling, and d) effective performance measures generation.

References

[1] L.P. Chien and J.W. Rozenblit, An Environment for Automatic System Performance Evaluation, Proceedings of the 1993 Winter Simulation Conference, pp. 582-588, Los Angeles, CA December 1993.

[2] J.R. Jackson, Networks of Waiting Lines, *Operations Research*, Vol. 5, pp. 518-521, 1957.

[3] P.J. King, *Computer and Communication Systems Performance Modelling*, Prentice Hall International (UK) Ltd., 1990.

[4] L. Kleinrock, *Queueing Systems*, Vol. I, John Wiley & Sons, Inc., 1975.

[5] A.M. Law and W.D. Kelton, *Simulation Modeling and Analysis*, McGraw-Hill, Inc., 1982.

[6] M.F. Morris and P.F. Roth, *Computer Performance Evaluation: Tools and Techniques for Effective Analysis*, Van Nostrand Reinhold Company, 1982.

[7] T.G. Robertazzi, *Computer Networks and Systems: Queueing and Performance Evaluation*, Springer-Verlag New York, 1990.

[8] J.W. Rozenblit, Experimental Frame Specification Methodology for Hierarchical Simulation Modeling, *International J. of General Systems*, Vol. 19, No. 3, pp. 317-336, 1991.

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

[10] B.W. Stuck and E. Arthurs, *A Computer and Communications Network Performance Analysis Primer*, Bell Telephone Laboratories, Inc., 1985.

[11] A.S. Tanenbaum, *Computer Networks*, 2nd Edition, Prentice-Hall, Inc., 1988.

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

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