Proceedings of the 1994 Winter Simulation Conference
ed. J. D. Tew, S. Manivannan, D. A. Sadowski, and A. F. Seila

MODEL DESCRIPTION IN THE INSYDE SIMULATOR FOR EVALUATING LARGE-SCALE COMPUTER SYSTEM PERFORMANCE

Toshio Komatsu
Hirofumi Wakayama

NTT Network Information Systems Laboratories
Nippon Telegraph and Telephone Corporation
Yokosuka-Shi, Kanagawa, JAPAN

Junro Nose

Productivity Innovation Promotion Headquaters
NTT Software Corporation
Yokohama-shi, Kanagawa, JAPAN

ABSTRACT

The INSYDE simulation tool was developed for use in the performance evaluation of large, complex computer systems. This simulator is an interactive graphical type tool in which model data is input in the form of diagrams and tables, and the results are also output in the form of diagrams and tables. This paper focuses on the method used by INSYDE to achieve highly powerful model description, an important issue with respect to practical use. The practicality of INSYDE is demonstrated by applying it in the evaluation of a number of complex, large-scale systems that are in the design stage.

1 INTRODUCTION

The increasing sophistication and complexity of computer systems is accompanied by a demand for accurate and timely means of predicting the extent to which performance targets will be achieved, estimating the limits of processing capability and identifying system bottlenecks over the entire life cycle, from system planning to the operation phase. Simulation is in increasing demand as one such effective method (See Komatsu and Nose 1991). We developed the INSYDE interactive graphical performance evaluation tool to allow easy evaluation by means of simulation. With INSYDE, modeling data can be described in the form of diagrams and tables.

We focused on practical use in the development of this system. While, of course superior operability, debugging functions and editing functions were implemented, emphasis was placed on achieving high descriptive power. The main issues were what degree of logical complexity could be expressed and to what extent intricate conditions could be simplified.

For the description of logically complex conditions, we are employing the following approach.

(1) Provide support for definition terms that can describe hardware and software resources in detail.
(2) Provide support for 56 types of nodes for describing the processing flow in the internal operation of computers.
(3) Make it possible to access global information, which is shared by the entire system, and local information, which is particular to each transaction, when deciding on conditions.
(4) Allow conditions that cannot be represented in diagram and table form to be defined by user programming.

For simple description of intricate conditions, we are taking the following approach.

(1) Allow indirect specification of access paths by files by using file path structure diagrams and a file table that indicates the file storage conditions.
(2) Node parameters can be specified by variables as well as by constants.
(3) Variables whose value can be decided uniquely before the transaction appears in the processing flow are defined in table form.

The rest of this paper is organized as follows. Section 2 explains the special processing characteristics of computer systems; section 3 provides a brief general description of INSYDE; section 4 presents various methods of improving descriptive power; section 5 presents examples of application in the evaluation of actual systems.

2 SPECIAL CHARACTERISTICS OF COMPUTER SYSTEMS WITH RESPECT TO PERFORMANCE EVALUATION

From the modeling point of view, computer systems have the following special properties with respect to the application of INSYDE.

(1) The system consists of a very large number of hardware resources, including processors, file storage devices, communication channels and terminals.

(2) The system's hardware resources contain from tens to hundreds of software resources, such as tasks, tables and files.
(3) The transactions in the system are highly varied (from tens to hundreds of types) and occur in large numbers (from thousands to hundreds of thousands of items per hour).
(4) There are processes that are special features of computer systems, such as interrupts, exclusion control, inter-task communication and synchronization.
(5) The resources accessed, the processing amount and other such factors vary from transaction to transaction, even for transactions of the same type and same processing flow.

From the above, we can see that there is a need to attain a model description capability that can solve the problems of scale, logical complexity and intricacy of description. Interactive graphical simulators have been reported in various application fields. In the communication network field, there are several systems (See Bharath and Kermani 1984, Marsan et al. 1990, and Dupuy et al. 1990). However, none of these can easily describe a detailed model for the internal processes of a computer system. While there are systems such as CAB in the computer field (See Okada and Tanaka 1991), these systems cannot provide a simple description for the special types of internal processes found in computer systems, such as interrupt processing and inter-task communication. Complex models are thus either impossible to describe or the description must be accomplished by user own coding. This is the key problem with respect to descriptive power.

3 OVERVIEW OF INSYDE

3.1 Functions

INSYDE consists of four blocks. The functions of each block are described below.
In the model definition information acquisition block (MDEF), the format of the model conditions input in diagram and table form is checked and converted to a format that is suited to the automatic generation of a simulation program. In the input of the processing flow, the operator selects the desired nodes from a node menu list on the screen, positions them as desired on the screen, and then sets the node parameters while referring to the guidance provided.

The automatic simulation program generation block (SGEN) generates a simulation program from the converted data output by MDEF. At the same time, it performs a logic check on the correspondence between the tables and diagrams.

The simulation execution block (SIML) executes the simulation program generated by SGEN. Time statistical data for all resources and each type of transaction and wait time statistical data are acquired automatically. For a single execution pass, statistical data can be acquired multiple times at fixed time intervals, which makes it possible to confirm the stability of the simulation. Also, detailed trace data, such as the transaction number and node parameters for each node are acquired so as to facilitate validation of the model. Transactions are often segmented during the flow of processing, so generation numbers and ID numbers are assigned to make it possible to distinguish them.

The statistical data editing block (EDIT) compiles the statistical data obtained in SIML and edits it for display in table or graph form. Also, to facilitate the retrieval of trace data, trace data can be searched for and displayed by using logical expressions involving the transaction number, the process flow name, the node name, the resource name and so on specified as search conditions.

3.2 Modeling Overview

An overview of the modeling process is illustrated in Fig. 1. In INSYDE, system operation is defined in the form of

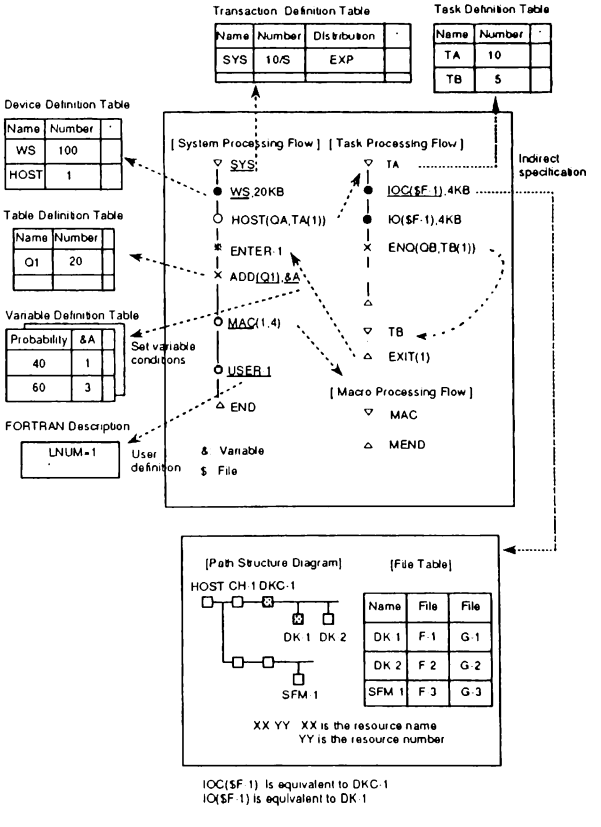


Figure 1: Modeling Overview

a flowchart representing the process flow, focusing on the flow of transaction processing. The direction of transaction flow is represented simply by the lines connecting the nodes. To make it easy to build, revise and extend a model, the conditions for resources and transactions and so on are defined in a diagram and table format, separated out from the process flow. Conditions that cannot be described with diagrams and tables can be defined by user programming. To make the processing flow easy to understand, comments can be inserted freely in the process flow description.

The process flow is described using the 56 types of nodes described in Table 1. Those types of nodes include one that users can define for themselves. INSYDE allows the processing flow to be described on three hierarchical levels: the system processing flow, in which the overall processing flow from transaction generation to completion is shown; the task processing flow, in which individual task processing is implemented; and the macro processing flow, which involves a group of processes that is to be repeated any number of times.

Table 1. Functions of Nodes for Processing Flow Description

Type of Node		Number	Functions
Time delay		8	These delay transactions with resources in the reserved or unreseved statud.
Control	Start, End	8	These indicate the beginning or end of processing flow. For termination, continuous restart and queuing to other processes is possible, except when the transaction is deleted.
	Branch	14	Unconditional branching and branching according to various conditions is possible. Conditions can be set for the work area of each transaction or variable, etc.
	Processing Transfer	9	These transfer processing between system processes and task processes or between task processes.
	Resource management	9	These handle resourcereservation and release. Multiple resources can be reserved or released simultaneously.
	Splitting and merging	4	These split transactions into multiple processes that may proceed in the same direction or in different directions. It is possible to specify that the split transaction be complete re-integrated or partially re-integrated.
User		1	This is defined by the user.
Macro		1	This calls a pre-defined macro process flow.
Measurement		2	These collect statistical data over a freely specfied measurement interval.

4. METHOD FOR INCREASING DESCRIPTIVE POWER

4.1 Describing Complex Conditions

4.1.1 Defining Task Structures

Concerning tasks, which are one of the software resources, various structures and control schemes have been proposed and implemented for the purposes of improving performance and facilitating program development and maintenance management. In order to make possible modeling that does not lose the special features of these structures and schemes, INSYDE is designed to allow detailed task definition, as shown in Table 2. In this way, the structures of task queue groups and task groups that have various characteristics, such as are shown in Fig. 2 for example, can be defined completely by means of tables.

Table 2. Main Task Definition Items

Item	Description
Task name	
Number of tasks	For each type of task
Interrupts	Whether or not there are forced interrupts for the CPU
CPU reservation priority	Priority order among tasks for reserving the CPU
Startup condition	Condition for starting service
Task Queue name	
Queue capacity	Task queue length
Queue selection rule	Rule for calling transactions from the task queue
Number of task reservations	The number of tasks that can be processed simultaneously
Task reservation priority	Priority order for reserving tasks

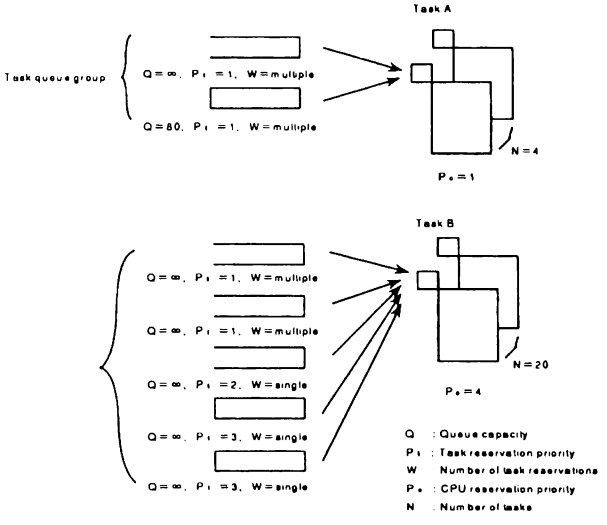


Figure 2: Example of Task Configuration

4.1.2 Special Processes of Computer Systems

Here, we give an example of describing control nodes, which represent the special processes of computer systems.

(1) Exclusion control: The LOCK node and UNLK node can be used to lock or unlock multiple resources at the same time. If a simultaneous locking fails, it is automatically judged whether or not simultaneous locking is possible for each of the target resources when any of them are unlocked.
(2) Interrupt processing: The PRE node performs interrupt processing, in which the task being executed is halted and the CPU is forcefully reserved. When the interrupt processing is finished, the interrupted task is automatically resumed.
(3) Inter-task communication: Information exchange between tasks is accomplished by the LINK and LON nodes. For each such communication, the initiator of the communication is managed on a per-transaction basis. Thus, as shown in Fig. 3, there is no need to specify the address for the response as a parameter for the LON node.

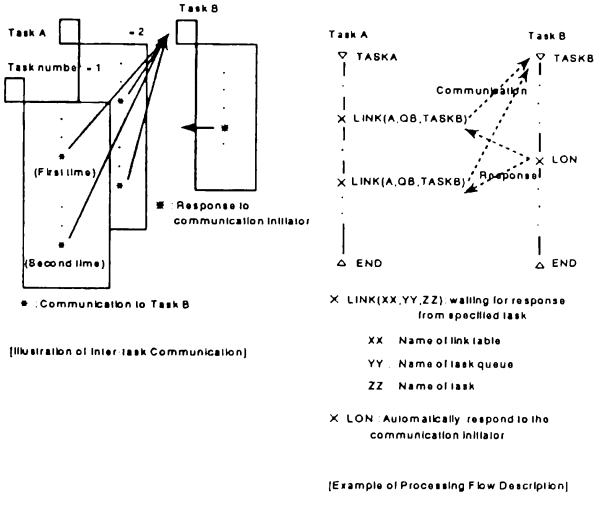


Figure 3: Example of Inter-task Communication Description

(4) Synchronization: As shown in Fig. 4, AND nodes are used to specify the synchronization of any number of split transactions. There is also another type of synchronization node, the OR node, which specifies that the first transaction that is reached is made effective, while the subsequent one is deleted.

4.1.3 Accessing System Information

System information can be referred to at any time during a simulation by using the IF node. Various conditions can be specified for the referencing. Global information (common to the entire system) and local information (concerning individual transactions) can be accessed. Global information includes such data as the using multiplicity, and number of waits for each resource. Local information includes the contents of the user-definable work areas and so on. This sort of access to system information allows the specification of complex conditions. In the case shown in Fig. 5 for example, first it is determined whether or not the number of transaction queues at a particular resource has exceeded a threshold. If it has, execution is retried a specified number of times, after which the transaction is de-

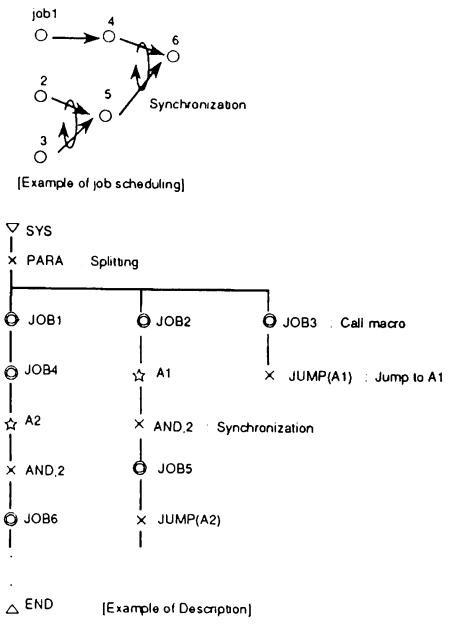


Figure 4: Example of Synchronization Description

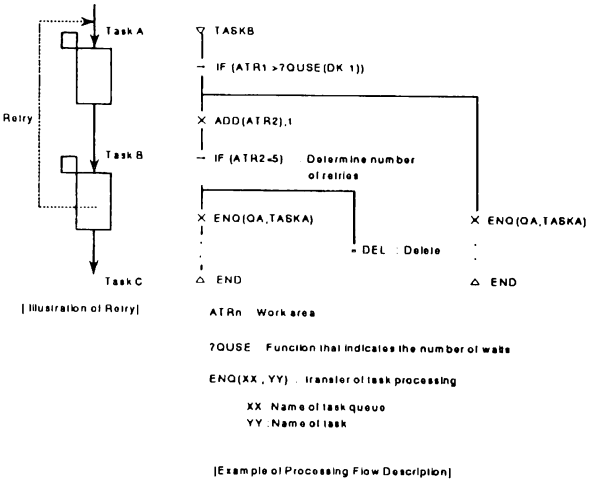


Figure 5: Example of Retry Description

leted.

4.1.4 User Defined Conditions

Conditions that are difficult to describe with the standard diagram and table format can be defined by the user in FORTRAN. User definition is not limited to within the user nodes in the processing flow; user programming can also be incorporated into the parameters of each node, queue selection rule of resource definition table, and generation distribution of transaction definition table. In the user programming, functions for acquiring various types of statistics are available in addition to the ability to refer to the various informations managed by INSYDE while the simulation is running.

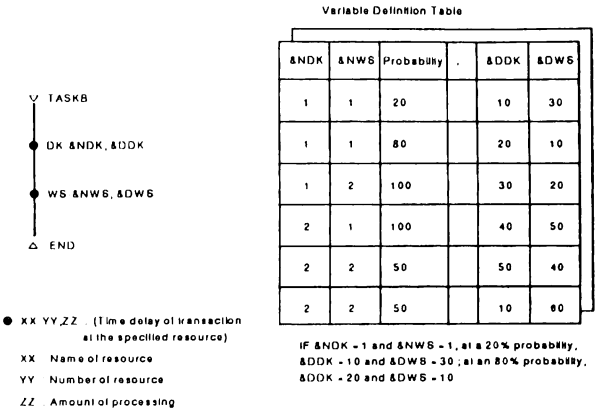


Figure 6: Example of Specifying Variables

4.2 Simplifying Intricate Conditions

4.2.1 Indirect Specification of Access Paths with Files

Files are frequently accessed in computer systems. This access requires specification of the access path to the file (the channel device, the file storage device, and so on) in the processing flow. Because this is difficult, the file access path is specified indirectly. This indirect specification is done by defining the path structure diagram, which represents the interconnections in the group of devices that make up the access path, and the file table, which represents the file storage conditions. With this approach, only the file name need be specified in the processing flow.

4.2.2 Using Variables in Node Parameters

If multiple processing flows are described individually when the processing flows are the same but the node parameters (device number, process amount, etc.) differ according to the transaction, then the description will be very large. This problem can be avoided by making it possible to define these kinds of parameters as variables. In this way, a single process flow description can be used for a given type of node.

4.2.3 Definition of Static Variables in Table Format

There are two types of node parameters: those that are determined when the transaction actually begins within the processing flow, and those that are uniquely determined at the time the model is defined. The former are called dynamic variables; their relationships with the execution conditions are defined by user coding. The latter are called static variables; their relationships with the execution conditions are defined in tables. A description example is given in Fig. 6.

5. OPERATION EXAMPLE AND EVALUATION RESULTS

5.1 Operation Example

INSYDE was used to evaluate three systems. System A is for a public network; system B is for a private network.

Both are large-scale systems for managing work ranging from application to maintenance. System C is a medium-scale system for centralizing work related to charges. The system conditions and the size of the models resulting from the evaluation are listed in Table 3.

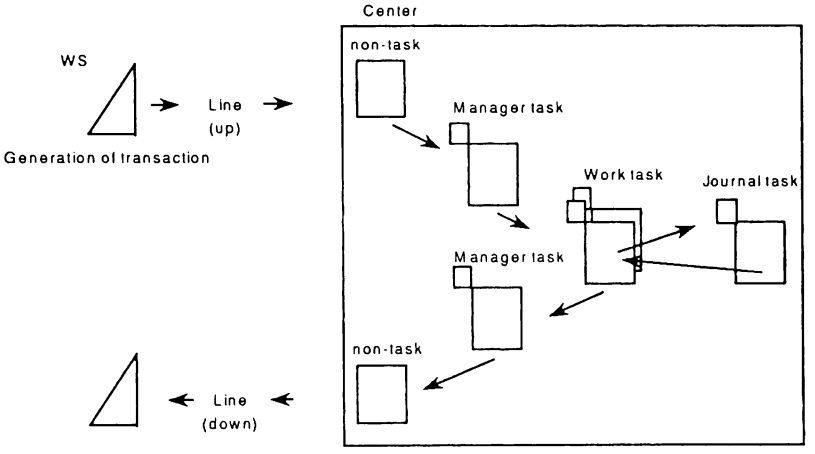
Table 3. Conditions and Modeling Scale for the Evaluated Systems

System Name	Main System Conditions			Model Size		F (1)/(2)
	Number of Transaction Types	Number of Devices	Number of Tasks	Number of INSYDE Nodes (Number of FORTRAN Steps(1))	Number of SLAM II Statements* (Number of FORTRAN Steps(2))	
A	8	20	25	675 (50)	1062 (342)	0.15
B	7	26	14	1026 (181)	2105 (455)	0.40
C	2	9	32	157 (0)	215 (125)	0

* Resource Statements defning resource names, multiplicity units and so on are not included

Part of the processing flow for system A for when INSYDE was applied is shown in Fig. 7. The figure shows the flow of processing for one of several transaction types. First, when transactions generated on a workstation arrive at the center via the communication line, the non-task interrupts the task being executed, performs reception processing, and then turns processing over to the manager task. The manager task divides up the transaction for processing by the work task. In the work task, DB1 and DB2 are simultaneously locked and processing is turned over to the journal task to accomplish database updating. The work task waits for the response from the journal task and writes the file in duplicate. Next, after executing lock release and other processing, the work task returns control to the manager task for termination processing. The manager task then returns control to the non-task for transmission processing, which sends the data to the originating workstation.

As can be seen from the figure, the processing flow is easily described and the description is easily understood.



[Flow of Transaction Processing]

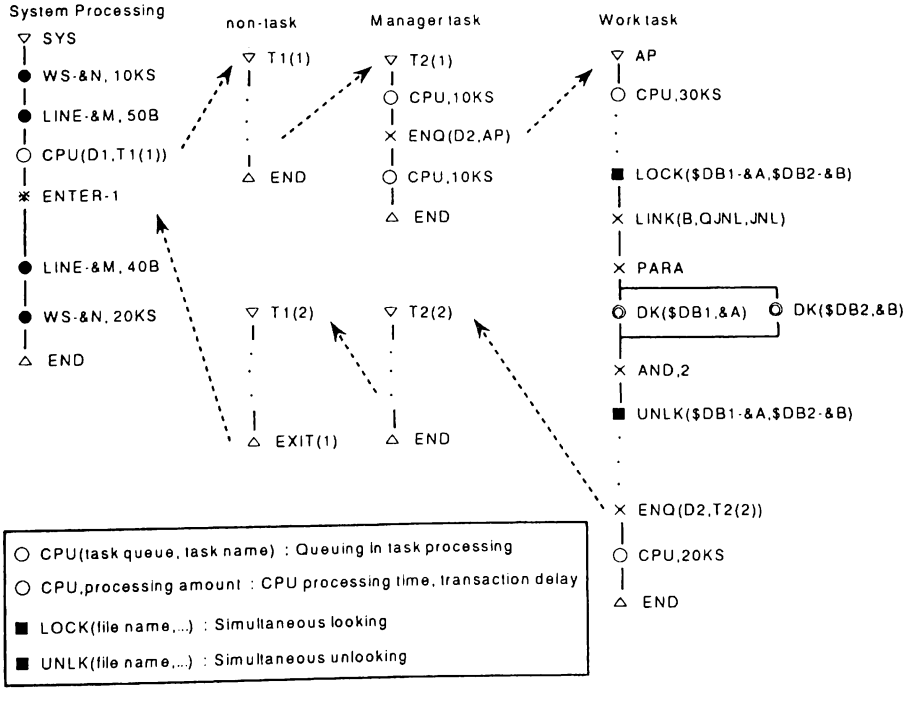


Figure 7: Example of Processing Flow Description

5.2 Evaluation Results

Performance with respect to intricate conditions can be qualitatively inferred from the various descriptions already presented, so in the following we discuss performance with respect to logically complex conditions.
Generally, the difficult point in the evaluation of performance versus the degree of complexity is that there is no clear measure for this purpose. Here, we will focus on the SLAM II simulation language (See Pritster 1986). Complex conditions that cannot be expressed using the standard statements provided by this language are written in FORTRAN. Therefore, we assume the amount of FORTRAN code used when the system under consideration is described with SLAM II to be one measure of the complexity of that system. The complexity of each system evaluated is given in Table 3. The main items written in FORTRAN are simultaneous locks, specification of resource numbers, and so on. The models for each system were respectively constructed by different persons.
(1) For each of the systems that have the degrees of complexity listed in Table 1, the system conditions could be described as they are, without approximation on a rough level.
(2) At this time, the following equation is used as a measure of the extent to which the description of complex conditions can be simplified.

$$F = N(I) / N(S)$$

where N(I) is number of FORTRAN steps used with INSYDE, N(S) is number of FORTRAN steps used with SLAM II.

Although the precision of the system conditions in question, the person creating the model, and so on are factors, we can say that F indicates the degree of simplification of complex condition description: the smaller the value of F, the greater the simplification. For the three systems evaluated here, F ranges from 0 to 0.4. This is primarily because the special computer system processes, such as locking, unlocking, and static variables, can be described with standard INSYDE expressions, whereas they must be coded in FORTRAN in SLAM II. The conditions written in FORTRAN in INSYDE were primarily specifications for dynamic variables.

We can thus conclude that while INSYDE cannot describe cases in which the transaction priorities are not fixed but change dynamically, it does achieve a generally high descriptive power.

6 CONCLUSION

We have described the INSYDE simulator developed for evaluation of the performance of computer systems, focusing on the method for achieving a powerful model description capability. INSYDE makes it possible to describe logically complex conditions and to simplify intricate conditions by means of techniques such as detailed resource condition definition tables, copious types of nodes for describing the flow of processing, introduction of variables into node parameters, indirect specification of access paths by means of files, table definition of static variables, and user definitions.

Actual application of INSYDE to a number of large-scale, complex computer systems that are either in the design stage or in operation has confirmed that this simulator is adequately practical with respect to the results of performance evaluation support, model description power, operability, and the debugging and editing functions. We are confirming the models by using the tables and diagrams as they were input to INSYDE in a review of the system with the client that requested the evaluation.

Future work will aim at a tool that has high descriptive power for a general information communication network that integrates computer systems and communication networks.

ACKNOWLEDGMENTS

We are very grateful to all the people who cooperated in developing and improving INSYDE, and to the many persons who supplied us with actual system model data and valuable discussions.

REFERENCES

Bharath.K.K. P.Kermani(1984). *Performance Evaluation Tool(PET): An Analysis Tool for Computer Communication Networks*, IEEE JASC, Vol.SAC2, No.1, pp.220-225.

Dupuy.A, J.Schwartz, Y.Yemini, D.Bacon (1990). *NEST:A Network Simulation and Prototyping Testbed*, COMMUNICATION OF THE ACM, Vol.33, No.10, pp.64-74.

Komatsu.T, J.Nose (1991). *Evaluation Response Time for Network Service System*, IPSG SIG Notes, OS-50-9.

Marsan.MA, G.Balbo, G.Bruno, F.Neri(1990). *TOPNET:A Tool for the visual Simulation of Communication Networks*, IEEE JSAC, Vol.8, No.9, pp.1735-1747.

Okada.Y, Y.Tanaka (1991). *FES:A Toolkit System for the Development of Visual Interactive Simulators*, Trans.IPG Japan, Vol.32, No.6, pp.766-776.

Pritsker.A (1986). *Introduction Simulation and SLAMII*, John Wiley & Sons.

AUTHOR BIOGRAPHIES

TOSHIO KOMATSU is a Senior Research Engineer in the NTT Network Information Systems Laboratories. He received B.S. degree in 1972 and M.S. in 1974 in electrical engineering from Kyushu Institute of Technology. Since joining NTT in 1974, he has been active in R & D on computer systems hardware architecture and computer systems performance evaluation techniques. He is currently interested especially in modelling and analyzing techniques and tools for distributed computing systems. He is a member of the Information Processing Society of Japan.

HIROFUMI WAKAYAMA is a Senior Research Engineer, Supervisor in the NTT Network Information Systems Laboratories. He graduated from Kyoto University in 1970. Since joining NTT in 1970, he has been active in R & D on operating systems software, computer network architecture and protocols, and computer software and system evaluation. He is currently responsible in the R & D on systems evaluation techniques and tools. He is a member of the Institute of Electronics, Information and Communication Engineers of Japan, and of the Information Processing Society of Japan.

JUNRO NOSE is a Senior Manager in the Productivity Innovation Promotion Headquarters at NTT Software Corporation. He received B.S. degree in 1967 and M.S. in 1969 in electrical engineering from Kobe University. He joined NTT in 1969 and has been active in R & D on fault diagnosis technique, videotex communication system and systems evaluation techniques. He joined NTT Software Corporation in 1993 and is now responsible in the systems evaluation systems and services. He is a member of the Information Processing Society of Japan.