Proceedings of the 1991 Winter Simulation Conference
Barry L. Nelson, W. David Kelton, Gordon M. Clark (eds.)

SIMULATION OF COMPUTER SYSTEMS AND NETWORKS WITH MOGUL AND REGAL

Peter L. Haigh

High Performance Software, Inc.
4288 Upham Road
Dayton, Ohio 45429

ABSTRACT

MOGUL(TM) and REGAL(TM) are easy to use, powerful tools for simulation modeling of computer systems and communication networks. Using MOGUL and REGAL along with Wolverine Software's GPSS/H and PROOF, the user gets a complete environment for building computer system and network models, running simulation experiments, and evaluating results. Animation of the graphical output is supported. Graphical input specification of the processors, devices, and networks in a model will also be possible in an upcoming release.

1 INTRODUCTION

MOGUL (MOdel Generator with User Leadthrough) is a package for rapid generation of simulation models of computer systems and computer communication networks. REGAL (REport Generator And Lister) is a companion program to MOGUL for quick inspection of simulation results. When used with GPSS/H and PROOF, MOGUL and REGAL comprise a powerful set of tools for the system performance analyst. (GPSS/H and PROOF are products and trademarks of Wolverine Software Corporation)

MOGUL and REGAL are available from High Performance Software, Inc. on a personal computer DOS platform and on selected UNIX platforms (Unix is a trademark of ATT).

2 OVERVIEW

MOGUL provides an expandable repertoire of processors, peripheral devices, and communication links. The user picks the desired type of object for each processor, device, and link in the model using a series of menus. Activity flow in the model is specified using an easy to use interactive editor, complete with on-line help for the high-level activity description language. The user indicates the amount of simulation time for the startup period and the run time. When the model definition is complete, MOGUL generates a complete GPSS simulation program. The GPSS/H compiler is used to compile and execute the simulation. REGAL may then be used for a quick inspection of selected statistics from the simulation output. An animation of the simulation may also be run on a personal computer.

Using MOGUL and REGAL for simulation modeling of computer systems and networks gives the user the following advantages:

- *Productivity.* MOGUL is extremely easy to use. The productivity of the MOGUL/REGAL process, compared to the task of writing a simulation program using a general purpose simulation language, is on the order of fifty to one (Bornhorst and Haigh, 1986).
- *Predefined Objects.* No programming is necessary to model processors, devices, and communication links. Simulation code is included.
- *Communication Protocols.* Most link level protocols are provided with the package, including BiSync, SDLC/HDLC/X.25, CSMA/CD (Ethernet et al.), Token Ring, SCSI Bus, and several others.
- *High-level Language.* The user codes activities in the model using a powerful high-level language, which is specifically designed for simulating computer system components, including software.
- *User Friendliness.* The menus and screens are human engineered, with on-line help and error checking during user input.
- *Reliability.* The simulation code for the standard system components has been used and tested in many models. This makes MOGUL generated models highly reliable.

- *Animation.* An animation of the simulation output may be viewed using Wolverine Software's PROOF. Animation is useful for model verification (debugging), demonstrations, and greater understanding of the simulation systems.
- *User-Defined Objects.* The user may extend or modify the repertoire of processors, peripheral devices, and communication links. All definitions are in an editable text file.
- *User Simulation Code.* The user may code processes is GPSS which are not supported by the MOGUL primitives. This code may then be included with a MOGUL generated model.
- *Any Level of Detail.* Although a model my be defined at a high level, the path definition language, coupled with the ability to add GPSS code, permits any level of detail to be built into a model.
- *Language Extensions.* The high-level language may be extended to include new statement types created by the user.

3 THE MOGUL WORLD VIEW

MOGUL views the world (to be simulated) as consisting of *processors*, *peripheral devices*, and *communication links*. These are types of objects whose behavior may be described by parameters. The parameters represent time delays for such things as transfer rate, latency, etc. and flags to indicate such things as "is it a disk?", "does it have a cache?", etc. The repertoire of objects is described in a text file, which may be edited by the user. The code to simulate these three basic object types is included in the package.

In addition to these three types of objects, there are constructs called *activity paths*. The activity paths consist of statements in a high-level language. The activity specified by each statement may be such things as "use 3 msec of cpu time", "send a message to cpu #3", "write 512 bytes to disk 2", etc. There is no limit (except the host machine's memory) to the number of activity paths the user may define, or to the length of any path.

The final element to complete the world view is a dynamic object called a *message*. When the simulation begins, messages begin flowing down one or more of the activity paths. As a message encounters a statement, it causes that activity to be simulated. There may be any number of messages in the model at a given time. Messages may be created automatically, where the user specifies the mean of a Poisson interarrival process. Messages may also be created by other messages by executing a "spawn" statement.

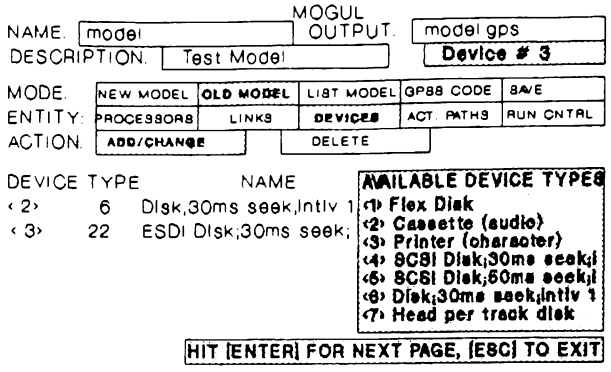

Figure 1: MOGUL Menu—Defining Devices

4 BUILDING A SIMULATION MODEL

From the MOGUL top menu, the user may define a new model, print a hard copy of the current model, produce GPSS source code, or save the current model. The menus begin user interaction near the top of the screen, and progress downward as menu choices are selected. Thus, a trace of the choices made to arrive at the current menu screen is evident.

4.1 Objects

A repertoire of basic object types is provided from which the user builds the simulated system.

4.1.1 Selecting the Objects

Menus are provided for instantiating the processors, peripheral devices, and communication links to complete the block diagram of the simulated system. Figure 1 shows a replica of the display seen during the definition process. In the example, the user is modifying an existing model [OLD MODEL]. The device definition menu has been selected [DEVICES]. The user is adding a new device to the model or changing an existing device definition [ADD/CHANGE]. Near the top, the current or last device to be affected by user input is displayed [DEVICE #3]. Similar menus are used for selecting processors and communication links.

4.1.2 Graphic Object Definition

With Release 2, the user may alternately choose to lay out the system diagram graphically, i.e., using PROOF and a mouse. The PROOF diagram can then be interpreted by MOGUL and incorporated into the

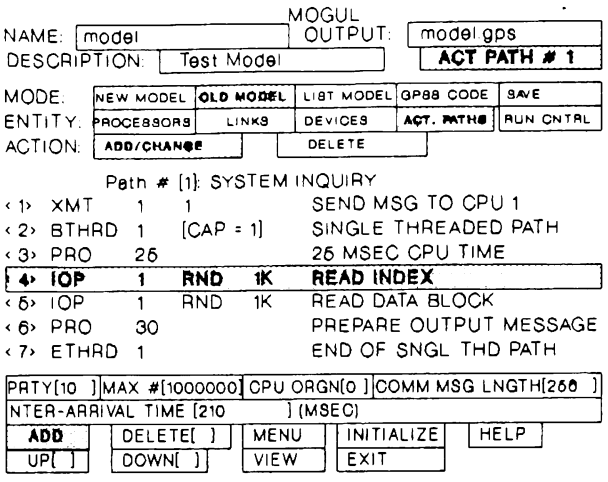

Figure 2: MOGUL Menu—Activity Path Entry

model. The user may then request a trace file to be produced by the simulation to drive the animation.

4.2 Defining Activity Paths

A replica of the screen display when editing activity paths is shown in Figure 2. The top part of the screen is similar to the object definition menus. We can see that the user is modifying an old model called "Test Model" and that Activity Path 1 is currently being edited [ADD]. Path statement 4 is highlighted, indicating the current cursor position. Additions or deletions would occur relative to this point.

From this menu, the user may move the cursor, add or delete path statements, page forward or reverse, view a larger portion of the path in an expanded window, set the initial conditions for a message entering the path, or obtain help for the activity description language. Automatic syntax checking is performed as new statements are added to the path. Additional error checks are performed when the GPSS code is generated to head off possible run time errors.

4.2.1 Language Flexibility

The design of the MOGUL high-level activity-description language permits a great amount of flexibility. A complete description of the language is available from the author.

Registers One hundred general purpose registers are provided, with a full set of arithmetic operations. These may be used for flow control through testing and skipping operations. Register values may also be specified for any operand requiring a numeric value. This permits dynamic indirect addressing. For example, "PRO R5" means process (use the cpu) for the number of milliseconds in Register 5.

User Code MOGUL will accept any statement with the op-code USRx, where x is an integer, without checking the syntax. To cause a message to branch from an activity path to a user written GPSS routine labeled USR23, for example, one needs only to enter USR23 as the statement in the activity path. Parameters may be passed to the user routine and the message may be returned to the activity path at the user's discretion.

5 EXTENDING MOGUL

The user may extend MOGUL by adding processor, peripheral device, and communication link types. In addition, the user may create new statement types for the activity description language used for the activity paths. User-written model segments may also be included with a MOGUL generated model.

5.1 Defining New Object Types

The types of processors, peripheral devices, and communication links are defined in a text file. The definitions consist of time values for various operations characteristic of the object type. Processors, for example, are characterized by the cpu execution time required by device drivers and interrupt handlers. A peripheral device may have a start up time, latency, data transfer rate, etc. To create new object definitions, or modify existing definitions, the user may edit the definitions file with any text editor. Utiltiy software is provided with MOGUL to assist the user with this process.

5.2 Defining New Protocols

If a protocol to be added is a variation of an existing protocol (i.e., it differs from an existing protocol merely by a difference in parameter values), it may be added to the prtocol list by editing the definitions file. If the protocol logic itself is new, then code must also be added. By modifying the definitions file and following the coding interface rules, the user may completely integrate new protocols into the MOGUL repertoire.

5.3 Extending the Language

The syntax for each statement type, or verb, is contained in the definitions file. New statement types may be added to the repertoire by editing the file and following the definition format. An entry specifies an operation code, the minimum and maximum number of operands, default operand values, expression types permitted for each operand, and text for on-line help.

6 REGAL

REGAL may be used to examine the statistics from a simulation run. REGAL parses the GPSS/H output file and generates a set of structures containing the major statistical information. The user may view resource utilizations, queue statistics, various model variables (i.e., GPSS savevalues), and table data. This process allows one to quickly peruse the results in a convenient screen format, rather than the wrap-around line printer format. REGAL can also produce coarse distribution graphs of response time or transit time data.

6.1 Snapshots

REGAL can produce a graph of a model variable over time by taking the value from a series of snapshop statistical reports produced during a run of a model.

6.2 Multiple Run Graphs

REGAL can produce a graph of a model variable as a function of the simulation run. For example, assume a series of simulation runs of a model is made. With each run, the processing load on the modeled system is increased. The simulation outputs are saved in file1, file2,..., etc. REGAL can then pick a model variable value from file1, file2, etc., and graph it. The result is the variable value as a function of processing load.

7 ANIMATION

Animation is a powerful tool for analyzing simulation results. Each change of state of a system may be observed, if desired. The primary applications are 1) debugging and verifying the model, 2) gaining a better understanding of the system behavior, and 3) demonstrations and presentations.

Several standard system configurations are available for automatic display. The real power of the animation feature, however, is the ability of the user to draw the system diagram using PROOF. Some objects which may be animated are processors, peripheral devices, communication links, queues, and response times. The user selects the objects to be animated, and MOGUL automatically displays relevant information according to the class of object (i.e., processor, communication link, etc.).

8 CONCLUSION

MOGUL and REGAL, in conjunction with GPSS/H and PROOF, provide a powerful and productive environment for simulating computer systems and networks.

REFERENCE

Bornhorst, E, and P.L. Haigh. 1986. A user friendly environment for simulating computer systems. In: *Modeling and Simulation on Microcomputers*, ed. C.C. Barnett. 1986 Society for Computer Simulation Multiconference.

AUTHOR BIOGRAPHY

PETER L. HAIGH is founder and president of High Performance Software. He has been active in the simulation community for many years and has authored papers on simulation and computer system performance topics. He was General Chairman of the 1988 Winter Simulation Conference.