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

THE RESEARCH QUEUEING PACKAGE MODELING ENVIRONMENT (RESQME)

Kow C. Chang
Robert F. Gordon
Paul G. Loewner
Edward A. MacNair

IBM Thomas J. Watson Research Center
Yorktown Heights, New York 10598, U.S.A.

ABSTRACT

The Research Queueing Package Modeling Environment (RESQME) provides a graphical environment for constructing, solving, and analyzing the results of extended queueing network models of resource contention systems. It has been used to improve the performance of existing and planned systems in such application areas as computer systems, communications networks, manufacturing processes, transportation systems, customer service facilities.

RESQME provides the capability to specify a queueing model by drawing a network diagram and attaching attribute information to each object in the diagram. It then evaluates the model using its general-purpose, discrete event simulation software and produces graphical and tabular performance results, along with animation on the original model diagram. Confidence interval methods are incorporated to insure that the results of a simulation meet the desired level of accuracy.

1 INTRODUCTION

Manufacturing lines, communication networks, computer systems, and transportation systems are examples of systems which are sufficiently complex and expensive that carefully developed models are needed in order to understand and improve system performance. In these systems, *contention* for the use of system resources is the primary factor affecting performance. Thus, queueing network models are typically employed for modeling system behavior. These models can be broadly divided into "traditional" queueing network models (solved analytically) and "extended" queueing network models (solved through simulation) (Lavenberg 1983). In both cases, the performance model contains queues and "nodes" (which represent the system resources), and jobs (which represent the objects in the system which contend for the use of these resources). The purpose of the model, then, is to analyze the effect of contention on the flow of jobs through the system.

RESQME (Kurose et al. 1986; R.F. Gordon et al. 1986, 1988, 1991; K.J. Gordon et al. 1991) provides the capability to specify and modify a queueing model by drawing a network diagram and attaching attribute information. It then evaluates the model using its general-purpose, discrete event simulation software and produces graphical and tabular results, along with animation on the original diagram. Confidence interval methods are incorporated to insure that the results of a simulation meet the desired level of accuracy.

RESQME runs under OS/2 on the PS/2 and under AIX on the RS/6000, as well as cooperatively between the PS/2 and a host VM system. The latter option provides the flexibility to evaluate models either in standalone mode on the workstation or cooperatively on the host. RESQME can also be accessed on a LAN.

The modeler controls the modeling process with RESQME by selecting menu items and directly manipulating icons on the display. Information about the model is provided at two levels. At the most visible level is the graphical view—the network diagram, its elements and submodels, animated job flow, and output charts. At a second level, there is attribute information for each graphical object. For example, a queue icon object has attributes consisting of its name, type of queue, queueing discipline, service time, etc. Similarly, an output chart object has attribute information defining its chart type (line, bar, histogram), colors, numerical values. The network diagram has attribute information, such as the model name, solution method, parameter names, run

control limits. The attribute information is presented to the user in a pop-up window whenever the modeler chooses to look at or modify it.

Interaction with RESQME divides broadly into three tasks which comprise the modeling process: Create/Edit, Evaluate, and Output Analysis including Animation. Within any of these tasks, the modeler can adjust the viewing plane by panning, zooming, layering to view the hierarchy of submodels, or locating objects by name on the model diagram.

During the Create/Edit task, a model is constructed by selecting icons representing the RESQ (R.F. Gordon et al. 1991) modeling primitives from a palette and placing them on the modeling "canvas". Textual attributes associated with an icon are then specified in a context-sensitive, pop-up window. Job flow among the elements within a model is specified simply by connecting the desired icons by pointing with a mouse and filling in the conditional and/or probabilistic routing specifications in the attribute window.

In addition to the elementary icons in RESQME, the modeler can draw his or her own icons and associate them with a subnetwork. Models can then be built hierarchically by linking existing submodels together to create a new model. RESQME thus supports the reuse of code through submodel libraries and the composition of icons into reusable higher-level icons.

The second task in the modeling process is to Evaluate the model. RESQME allows the modeler to set up multiple runs by varying parameter values and then to execute the series of simulation runs. The modeler has the option to evaluate smaller models and pilot runs standalone on the workstation and larger models on the host.

The third task is the Output Analysis task. RESQME provides a plotting package to graph the collected performance results. Performance results related to a specific icon or routing chain are selected simply by pointing to that icon or chain and then selecting the performance measure(s) of interest from the pop-up window of available measures. The modeler can analyze performance measures from one model run or across runs, for one node or for many nodes, and view the results in a number of different plotting options. Animation of the job movements through the network diagram is provided after the simulation run, showing job transitions between the nodes of the diagram along with queue lengths. The modeler can follow the movement of individual jobs through the submodels.

Finally, we note that RESQME was specifically designed to be used by both novice and experienced modeler. The novice modeler can play through tutorial scripts which demonstrate (via animated examples) various aspects of the system. We have built a large number of tutorials using the record and playback facility in RESQME. The tutorials provide a mechanism for teaching new users about simulation methodology, the use of RESQME, and the structures of RESQ. They also provide the expert with a mechanism to design customized, animated tutorials in order to, for example, document or explain a model that has been developed. For the expert, RESQME contains features which help expedite the modeling process. For example, there are "modes" of operation which permit easy repetition of the same task, library capabilities to share submodels, journaling functions, and confidence interval run control. For both novice and expert, there are libraries of submodels that can be used to construct models, default values for appropriate items, and scrollable predefined responses for many prompts.

2 THE CREATE/EDIT TASK

2.1 Specifying a Model

In RESQME, the performance modeler constructs a model by (1) selecting icons to represent the system resources and placing them on the modeling canvas in meaningful relative positions, (2) specifying attributes for each icon so that it represents the behavior of the corresponding system resource, and (3) interconnecting the icons on the canvas with paths which represent the possible flows of jobs and control in the system. Different modelers may wish to perform these actions in different sequences or may mix these sequences, building up resources and routing in the model diagram section by section. RESQME accommodates all such strategies, as well as the inevitable breaks in strategy which arise from false starts, editing, and debugging.

The modeler enters an icon into the model by picking it from the icon palette and placing it in the modeling area of the screen. For ease in later attachment of routing connections, the icon can be placed on the modeling canvas in any of the four cardinal orientations. When the icon is placed, RESQME displays a context-sensitive form (described below) for specifying the icon's attributes. The modeler may fill in attributes at this point, or may escape and leave them to be filled in later. Figure 1 shows a snapshot of RESQME during the

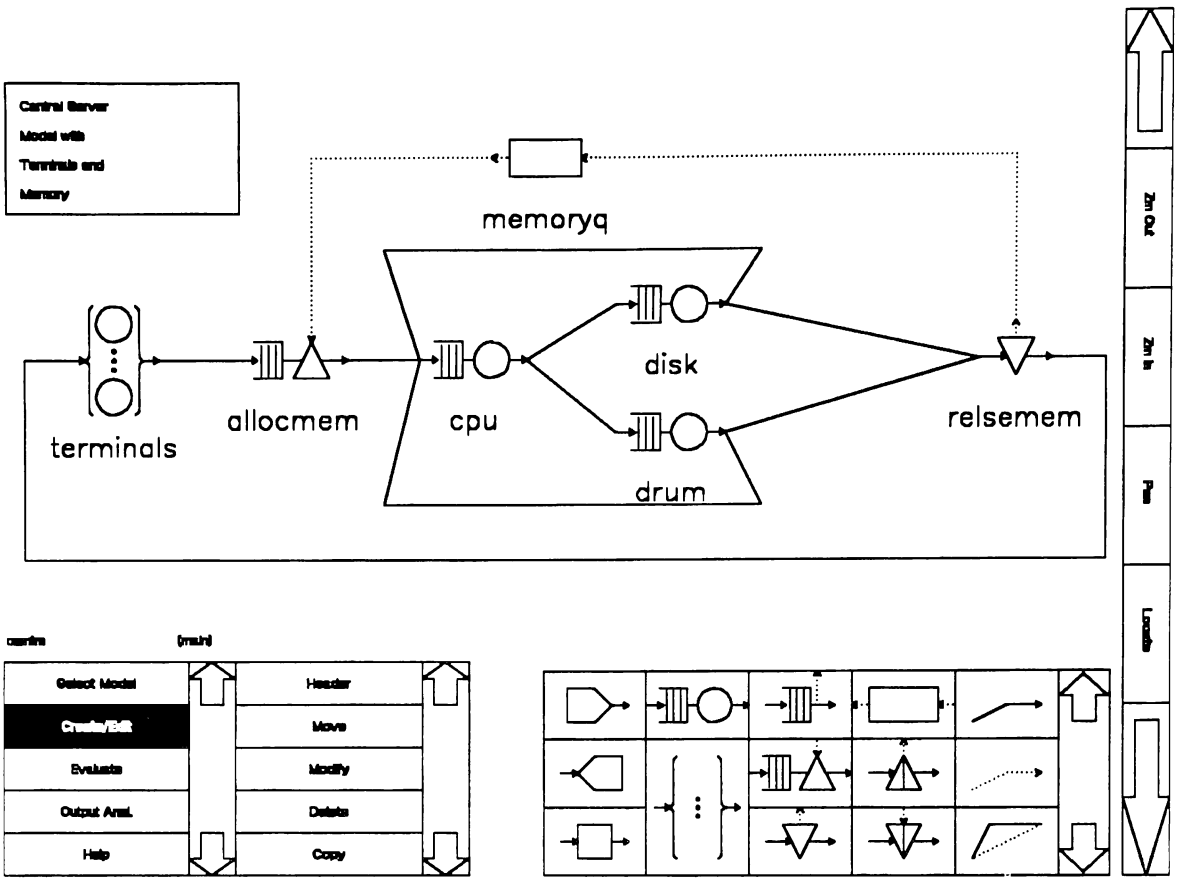

Figure 1: Model construction in RESQME

model creation process. The queue icons used to represent the CPU, memory, terminals, and I/O devices in the central server model are shown on the modeling canvas.

Once an icon has been placed on the modeling canvas, the modeler can Move, Delete, Replace or Copy the icon, or Modify its attributes. Copying an icon from the modeling canvas differs from Adding a new icon from the palette in that all the attributes of the copied icon except for its name are replicated in the new icon. If the modeler Moves an icon which has routing connections associated with it, the connections stretch or contract to link the icon in its new position and orientation. Commands such as Copy, Move, Delete, and Modify are *modal* in nature in that, once chosen, the command remains highlighted on the menu, and the modeler can perform the same operation on several icons in sequence without re-selecting the command. The use of modes has proven extremely useful in reducing the number of keystrokes needed in editing a model.

The icon attributes (such as its name, associated distribution (if any), priority information, etc.) are specified textually in RESQME using context-sensitive *forms,* which help reduce the need for the modeler to know the syntax of the underlying RESQ language. Figure 2 shows such a form. Attributes are entered in fields following each colon. The various commands available for manipulating the form are indicated at the bottom of the figure. The forms are context-sensitive in the sense that the specification of certain attributes may cause the remainder of the form to change dynamically. For example, if a priority queueing discipline is chosen in Figure 2. (by scrolling through a list of available queueing disciplines until the appropriate priority discipline is shown following TYPE:), additional lines for entering priority information will immediately appear.

As the modeler enters an attribute specification, the entry is parsed. If the parsing reveals any errors, the icon is colored red and the erroneous reply is also shown in red with an error message; if not, the icon is colored green. This incremental parsing provides early feedback and reveals not only erroneous input but also inconsistent or missing information.

```
QUEUE: cpuq
TYPE: ACTIVE
SERVERS: 1
DSPL: PS
CLASS LIST: cpu
  WORK DEMANDS: .004_
SERVERS-
  RATES: .2
  ACCEPTS: all


Enter work demand for this class

1Help 2Return 3Select 4Dupl 5Del 6Insert 7Up 8Dn 9Top 0Bot
```

Figure 2: Attribute specification form (for an active queue)

2.2 Specifying Job Routing

The movement of jobs among the resources in the model is described by chains of routing connections. If there are several classes of jobs in the system, jobs from each class may be allowed different routes. Routing specification can be extremely time-consuming, particularly when large models are being constructed. For this reason, RESQME provides extensive routing capabilities, rather than simply the minimally-required capability of connecting one icon to another.

The modeler specifies each routing connection by pointing first to the FROM icon and then to the TO icon. In most situations, the modeler will want to enter a sequence of links in a given chain. Therefore, by default, the TO icon automatically becomes the FROM icon for the next link. It is, however, quite easy to override this choice and select a different FROM node for the next connection. Depending on the geometrical arrangement of the icons, a straight line may not be the ideal representation for the routing path. RESQME permits the modeler to tack down the path at several intermediate points before picking the TO icon. (The model in Figure 1 contains tackpoints in several of its routes.) Although these tackpoints have no semantic value in the model, they can be moved or deleted just like icons. It is also possible to insert a tackpoint into an existing line segment.

Most models contain one or more branch points—an icon from which a job may go to one of several other icons, based either on a probabilistic decision or on some simulation-dependent condition. To aid in the specification of such "1—N" routing, RESQME allows the modeler to build up a list of TO nodes by double-clicking on the first icon in the list, single-clicking on the second through N-1st, and double-clicking on the last one. The same procedure can be used to build up a list of FROM nodes for convergent N—1 or parallel N—N routing. Upon completion of a 1—N or N—N link, all N of the TO nodes become FROM nodes, and the modeler may continue with N—N or N—1 routing. Figure 1 shows an example of 1—N and N—1 routing. RESQME also permits tackpoints in 1—N, N—1, and N—N routes. When the modeler puts down tackpoints between the lists of FROM and TO nodes, the routes converge to the first tackpoint, a single path runs from tackpoint to tackpoint (a single line thus representing N different actual paths in compressed form), and the routes diverge from the last tackpoint. The N—1, 1—N, and N—N routing, tackpoints, and compressed routing have proved particularly valuable in specifying the job routing in large models.

As the modeler graphically specifies the routing, RESQME also presents a textual confirmation of the routing, complete with default probabilities for the routing control. When the modeler finishes the graphical part of the routing for a particular chain, there is an opportunity to edit these probabilities or change them to Boolean predicates. It is also possible to reenter any chain to insert additional routes, to MODIFY the routing predicates, or to MODIFY the line style or color of the chain links. The modeler can also DELETE individual links or an entire chain.

2.3 Hierarchical Modeling Using Submodels

Systems in the real world can be large and complex. This complexity can arise from a large number of components in the system as well as from detailed structure within individual components. In either case, the modeler will find it conceptually useful to visualize the system at different levels of detail. RESQME supports this multilevel modeling concept with hierarchical model definitions. The conceptual unit in the hierarchy is a submodel definition—a collection of RESQME primitives and/or other submodels and the routing connecting them to one another and to the input and output connections to the outside of the submodel.

Graphically, each submodel is defined on its own modeling canvas. This single definition may be parameterized, and can be invoked as many times as the modeler wishes. Since the model can be constructed in a multi-layered hierarchy by the nesting of submodels, a model's submodels are thus related to one another as the nodes of a tree. The Lyr Dn and Lyr Up menu commands in RESQME provide the functionality for traversing the tree and for the creation of new nodes at the desired places in the tree.

RESQME also permits the modeler to extract and save submodel definitions. The saved submodels comprise a "library" of definitions, from which the modeler can recall a desired submodel and insert it into the appropriate place in a new model. This feature also makes it possible to create submodels which describe the same component or subsystem to different levels of accuracy or detail or, perhaps, which model differently designed components performing the same function in different versions of the system under study. These library definitions can serve the modeler as interchangeable building blocks in the construction of the model.

2.4 User-defined Modeling Elements

The purpose of user-defined modeling elements is to make it possible for the modeler to work directly with modeling constructs which closely represent objects in the problem domain. The provision of such domain-specific icons reduces even further the effort required from the modeler to translate his real-world problems into and out of the descriptive form mandated by modeling software. For example, a higher-level object might be created to represent a robot or a conveyer system in a materials handling system, a controller or a specific minicomputer in a communications network, or a work cell in a manufacturing system. Like any submodel, the higher-level object can be parameterized, so that each instance of it can invoke the submodel with different argument values.

In RESQME, the capability to extend the basic RESQ elements is built upon the hierarchical submodeling capability described in the previous section. RESQME provides capabilities to create user-defined icons and to associate them with submodels, thereby creating higher-level objects. The modeler can then use these higher-level objects as well as the basic RESQ elements in constructing a performance model. The submodel underlying a higher-level object can be created by the modeler using both basic RESQ icons and higher-level objects, or it can be selected from application libraries created by others.

The modeler creates a new object by (1) creating a submodel (or selecting an existing submodel), and (2) drawing the new icon and linking it to the submodel. The two steps can be done in either order. We provide an icon-drawing package which allows the modeler to draw icons (thus creating the internal icon structure used by RESQME) by using line, circle, and polyline elements, and to edit these elements with move, copy, and delete commands. Existing icons can be selected as a base for new icons and modified, or the modeler can start with a blank drawing box. The icon drawing package allows the modeler to group the resulting icons into user-created icon palettes for a given model or application area. The modeler links a specific icon to a submodel by providing it the name of the submodel.

Within RESQME, user-created icons are selected and manipulated in the same way as the basic icons. If the modeler selects a model with user-created icons, those icons will be displayed in icon palettes in addition to the two built-in palettes which contain the basic RESQ icons. When the modeler selects a user-created icon and places it on the modeling canvas, RESQME checks whether the submodel is already in the tree structure of the model. If it is not, RESQME will search for the submodel description, and if it exists, will attach it to the model structure. If it does not exist, the modeler can layer to a new modeling canvas and create it. The underlying text attributes for the user-created icon are displayed based on the submodel to which it is linked. The attributes include the submodel type and any parameter variables. The modeler provides the parameter values for each instance of the higher-level object.

The modeler can view the underlying submodel network at any time by selecting the Lyr Dn item from the screen manager menu and pointing to the desired user-created icon. RESQME will then dis-

play the submodel network. The modeler can also view any of the submodels in the model structure and move back to the root by using the screen manager menu item Lyr Up.

2.5 Support for Large Models

It must be clear at this point that RESQME models contain both graphical and textual information, that large models will contain large amounts of both kinds of information, and that (in order to be useful) RESQME needs to make it easy for the modeler to switch back and forth between graphics and text management while keeping the context of the model in the modeler's view.

The graphics management contains two classes of functions: icon-oriented and screen-oriented. The icon-oriented functions include adding, deleting, moving, replacing, and copying individual icons and the individual line segments which comprise the routing paths. The screen-oriented functions include zooming in and out, panning, and locating a named icon. If we think of the model diagram existing in its own coordinate frame, independent of the monitor screen, then the icon-oriented functions alter the model diagram while the screen-oriented functions merely alter the window through which the modeler views the diagram.

The behavior of the icon-oriented functions has been described in previous sections. The screen-oriented functions, which are accessed through the screen manager menu along the right edge of the graphics screen (see Figure 1), are particularly useful for working with large models.

The functions Zm In and Zm Out, respectively, magnify and reduce the picture of the model on the modeling canvas by a preset factor, while keeping the view centered over the same point in the model. The modeler can modify the value of the factor by editing the user-profile file, which is read in when RESQME starts up, or by changing it on the system-options attribute template, which is accessible through a menu item on the second page of the main menu. Zooming out gives an overview of an entire model.

With each invocation of the Pan function, the modeler can move the model around the canvas in an arbitrary direction by any distance up to the diameter of the screen. This involves selecting the function from the menu, choosing an arbitrary point on the screen (which does not have to be associated with any icon in the model), and choosing the new position for that point. The model is redisplayed in the new position at the same scale as before the move.

The Locate function centers the viewing window over a named icon. The modeler selects the function, then types the desired name into a small window which appears in the lower right corner of the modeling area. The icon to be located may be anywhere in the current layer of the model. In effect, this permits the modeler to pan the model through an arbitrarily large distance.

As previously mentioned, the Lyr Dn and Lyr Up functions display the network at other nodes of the model tree structure. The display of each layer of the tree is independent of the others' screen management (panning and zooming).

3 THE EVALUATE TASK

The second task in the experimental process is to evaluate the model. RESQME allows the modeler to provide sets of parameter values for each desired run of the model and then to execute the series of runs locally or on the host computer.

Selecting the Evaluate main menu item causes the Evaluate task menu to be displayed. This menu provides the modeler with the commands to enter parameter values and execute the model. A model can be executed with different parameter values in a series of experiments.

If using the host system to execute the model, selecting the Execute item then uploads the model files to the host and issues the host command to run the model. In this case, it is the command to run RESQ with the model files. The host execution gives us the computing power to run large, realistic models.

The host execution is done in the background, so that the modeler can continue to work at the workstation on this or other models while the host is processing the model. This cooperative processing takes advantage of the host for the computation-intensive execution of the model and the workstation for the interactive graphics.

The modeler can check on the progress of the run at any time. When the model results are ready, the workstation will sound a beep to notify the modeler.

Alternatively, the evaluation of the simulation can be done directly on the workstation, spawning a job to run the simulation.

4 THE OUTPUT ANALYSIS TASK

RESQME supports the output analysis task by providing a general-purpose plotting package to graph the results. This task is integrated with the other tasks of RESQME again using the same model diagram as the interface. The plotting package is gen-

eral in that it can plot the output from any modeling program as long as the output is put in the form: performance measure identification followed by a vector of x and y values. When the modeler selects the Output Analysis menu item from the Main Menu, a Task Menu appears with the commands to specify the content and the form of the plot, and plot the resulting chart. Figure 3 shows the RESQME display with this task menu and an example plot of performance measures.

The modeler can analyze performance measures from one model run or across runs, for one node or for many nodes. He or she can plot many results on one chart or on different charts. Whenever confidence intervals are produced by a simulation, they are automatically displayed on the charts for those performance measures. The specific numerical data for each performance measure are also available as the underlying textual attributes associated with the chart contents.

Several functions are made available to the modeler to help analyze the data. For example, the modeler can produce a short run using one of the confidence interval methods of RESQ. The modeler can then select performance measures from this pilot run and ask for a confidence interval projection. A plot is then displayed that estimates the required run lengths to meet a range of desired confidence interval widths at a desired percent. In addition, the modeler can apply smoothing functions to the data or, if desired, plot one variable against another. For example, the modeler may wish to plot the mean queueing time for several runs of the model against the corresponding interarrival time parameter values for those runs.

In RESQME, the form of the chart (type of chart, axis intervals, color, position, etc.) is controlled by the modeler independently of the contents of the chart (values of performance measures). Thus, the modeler can view the same contents in different forms and/or use the same form of chart for many different plots. This separation of form from content gives the modeler flexibility in analyzing the results, and the modeler can easily tailor the form to the type of information to be plotted.

We feel that, just as the graphics on input has reduced the translation necessary between the modeler's view of the system and the lower-level specification requirements of the underlying model solution package, so too the output graphics has reduced the translation burden in the opposite direction, more clearly assisting the modeler in interpreting the results. This is especially true for large models. Graphics provides the best means to manage large amounts of data by visually presenting the data for easy comparisons, quick determination of statistical significance, and clear spotting of trends.

The modeler specifies the performance measures to display in a given chart by pointing to a desired node in the model diagram. A list of performance measures for the associated node pops up for the modeler to choose. The modeler can choose any number of these performance measures and point to any number of other nodes for selection in the same chart. The modeler can choose nodes from the main model or any submodel. Additionally, the modeler can point to the model name to choose performance measures associated with the whole model or to a routing link to select performance measures associated with a route. To handle the volume of performance measure data for large models, the output data are stored on disk in an indexed file. Only when the modeler selects a given performance measure is it brought into the PC memory.

Our windowing system allows the charts to share the modeling surface with the model diagram. Charts can overlap the model diagram and other charts. The charts can be positioned, moved, shuffled, and removed by direct manipulation. Each submodel layer has its associated charts which are independent of the charts on other layers of the model.

The model diagram, then, is the basis for the selection of output as well as the interface for model creation, and the modeler can view the output and the model diagram simultaneously. The modeler can explore changes based on his analysis of the output by directly modifying the model diagram, re-evaluating the model, and then viewing the next version of the performance measures.

The animation subtask (Aggarwal et al. 1989) is started from the Output Analysis task. The animation shows the movement of jobs and tokens and the change in queue lengths of the nodes in the model and in the submodel invocations. RESQME can display the animation for any selected invocation or, at the modeler's option, it can trace a given job as it moves up and down through the invocations of the model. In the latter case, the animation will automatically layer to each invocation that the job visits. During this tracing of a given job, the status at queues and the movement of other jobs and tokens for each displayed invocation are also shown.

5 SUMMARY

In this paper, we have described our efforts in designing and developing the Research Queueing

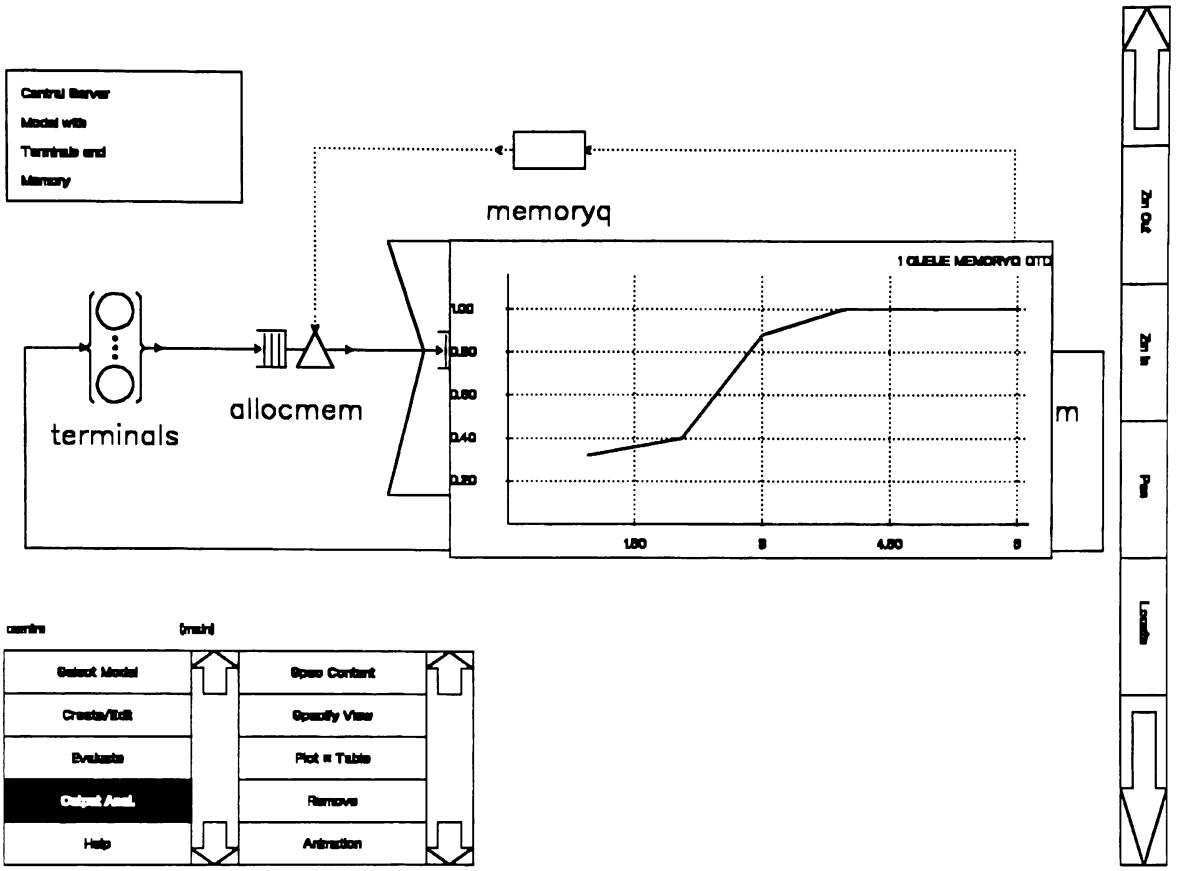

Figure 3: Example plot of performance measures

Package Modeling Environment (RESQME), a graphical environment for constructing, solving, and analyzing the results of extended queueing network models of resource contention systems.

RESQME is unique in that it provides a rich and extensible underlying "language" and a uniform graphical interface for constructing extended queueing network performance models of large and complex real-world systems. A single graphical interface is maintained throughout all aspects of the performance evaluation process. A model is constructed and modified by creating and editing the model diagrams and by entering textual information into context-sensitive forms. When a model is solved, the modeler specifies values for the model parameters for one or more solutions, and the model is sent to the host for evaluation. Under this cooperative processing protocol, the workstation is then free to be used for other purposes. When the solution is complete, the modeler can point to any desired node in the model diagram to specify performance measures which are then displayed in graphical and tabular forms. RESQME allows the modeler, through a consistent interface on the PC, iteratively to create the model, view the results of the analysis or simulation, revise the model based on the output, and compare results for families of models.

ACKNOWLEDGMENTS

We are grateful to the many colleagues and RESQME users who have helped improve RESQME over the years.

REFERENCES

Aggarwal, A., K.J. Gordon, J.F. Kurose, R.F. Gordon and E.A. MacNair. 1989. Animating simulations in RESQME. In *Proceedings of the 1989 Winter Simulation Conference*, ed. E.A. MacNair, K.J. Musselman and P. Heidelberger, 612-620. Institute of Electrical and Electronics Engineers, Washington, District of Columbia.

Gordon, K.J., J.F. Kurose, R.F. Gordon and E.A. MacNair. 1991. An extensible visual environment for construction and analysis of hierarchically-

structured models of resource contention systems. *Management Science* 37:714-732.

Gordon, R.F., P.G. Loewner and E.A. MacNair. 1991. The Research Queueing Package Version 3: Language Reference Manual. IBM Research Report RA-210, Yorktown Heights, New York.

Gordon, R.F., E.A. MacNair, P.D. Welch, K.J. Gordon, and J.F. Kurose. 1986. Examples of using the RESearch Queueing Package Modeling Environment (RESQME). In: *Proceedings of the 1986 Winter Simulation Conference*, ed. J.R. Wilson, J.O. Henriksen, and S.D. Roberts, 504-510. Institute of Electrical and Electronics Engineers, Washington, District of Columbia.

Gordon, R.F., E.A. MacNair, K.J. Gordon and J.F. Kurose. 1988. Higher Level Modeling in RESQME. In *Proceedings of the European Simulation Multiconference 1988*, 52-57. Nice, France.

Kurose, J.F., K.J. Gordon, R.F. Gordon, E.A. MacNair, and P.D. Welch. 1986. A graphics-oriented modeler's workstation environment for the RESearch Queueing Package (RESQ). In: *1986 Proceedings Fall Joint Computer Conference*, 719-728. Dallas, Texas.

Lavenberg, S.S., ed.. 1983. *Computer Performance Modeling Handbook*. New York: Academic Press.

AUTHOR BIOGRAPHIES

KOW C. CHANG is an advisory programmer in the Computer Science Department at the IBM Thomas J. Watson Research Center. His research interests include design and performance analysis of computer communications networks, applied probability, queueing theory and discrete-event simulation. He is a member of IEEE and an associate member of ORSA.

ROBERT F. GORDON is manager of modeling and analysis software systems at the IBM T.J. Watson Research Center. His research interests are in the areas of decision support systems and graphical environments. He is an adjunct professor in the Business Computer Information Systems and Quantitative Methods Department of Hofstra University.

PAUL G. LOEWNER was an advisory programmer in the Computer Science Department at the IBM Thomas J. Watson Research Center. His interests are in graphical systems, compilers, and mathematical analysis. He is a member of ACM, SIAM, AMS, MAA, IEEE CS.

EDWARD A. MACNAIR is a Research Staff Member in the Computer Science Department at the IBM Thomas J. Watson Research Center. His research interests are performance modeling tools and simulation output analysis. He is a member of ORSA and TIMS, and was *Proceedings* Editor for the 1989 Winter Simulation Conference.