Proceedings of the 1986 Winter Simulation Conference
J. Wilson, J. Henriksen, S. Roberts (eds.)

EXAMPLES OF USING THE RESEARCH QUEUEING PACKAGE MODELING ENVIRONMENT (RESQME)

Robert F. Gordon, Edward A. MacNair and Peter D. Welch
IBM Thomas J. Watson Research Center
Yorktown Heights, New York 10598

Kurtiss J. Gordon and James F. Kurose
Department of Computer and Information Science
University of Massachusetts
Amherst, Mass. 01003

ABSTRACT

The RESearch Queueing Package Modeling Environment (RESQME) is a system which provides an integrated, graphics-oriented, performance evaluation workstation environment for constructing, maintaining, revising and evaluating performance models of resource contention systems. In this paper we discuss examples illustrating its use and emphasize the iterative nature of the modeling process. Scenarios of model specification, selection and use of confidence interval methods, and model output analysis are introduced.

1. INTRODUCTION

We believe that graphics provides the most effective interface to investigate queueing problems, allowing the analyst to efficiently interpret a large amount of data by viewing a network diagram of the model and output graphs of its results (Browne et al. 1985, Gilbert and Kleinoder 1985, Healy 1985, Melamed and Morris 1985, Pegden, Miles and Diaz 1985, Sinclair, Doshi and Madala 1985, Standridge, Vaughan and Sale 1985a and 1985b). The analyst thinks in terms of a network diagram when constructing a model, when running a model, and when analyzing its results. He interprets the model's results by visualizing the graphs of its performance measures. Output graphs from pilot runs provide the analyst with needed information to design the experiment (for example, to determine run length and/or number of replications for statistical significance). Output graphs from detailed runs provide needed information for interpreting the results. Therefore, a graphical representation of a model combined with its output statistical graphs provides the basis for a single, uniform interface for all aspects of the modeling and performance evaluation process. We create such a modeling environment, called the RESearch Queueing Package Modeling Environment (RESQME), and provide the analyst with an integrated set of software tools for use throughout the modeling cycle. In this environment, the analyst views and manipulates the model directly through its graphical network representation. The analyst has access to both the model diagram and to the underlying attributes of the model's elements. Additionally, the analyst can view the model results graphically with flexible control of both the contents and form of the output. The results of an analysis may lead to further model refinements and a need to modify the model diagram. Presenting one environment gives the analyst the ability to refine, execute, and display the model at any time. The design and implementation of this environment, RESQME, is discussed in Kurose et al. (1986). An overview of RESQ, which forms the modeling language and solution component for this new environment, can be found in Chow, MacNair and Sauer (1985), MacNair (1985), MacNair and Sauer (1985), Sauer and MacNair (1982) and (1983), Sauer, MacNair and Kurose (1982a), (1982b), (1982c), and (1984).

There are three phases in the modeling process that are represented in RESQME. The first phase is model specification (and modification). In this phase, the analyst constructs and edits his queueing model. The second phase is model evaluation. In this phase, the analyst assigns parameter values and executes the model. The third phase is output analysis, in which the analyst displays the results of the model. The analyst can move freely between these phases until satisfied with the results of his experiment. RESQME also provides the tools for the overall control of the phases of the modeling process and the management of a data base of models and model versions.

The hardware for RESQME consists of a personal computer with a graphics display and a character display. The personal computer is connected to a main frame. The graphics display is used to specify and present the queueing network and the output performance measures. It is the main focus of the analyst and is used by the analyst to control the complete modeling process. The character display is used to specify and present attribute information about the network elements, the run parameters, and the content and form of the performance measure graphs. In this cooperative processing, the PC supports the creation of the queueing model, the specification of the run parameters, the display of the output results, the production of all graphics and the control of the modeling process; the main frame is used to execute the RESQ model, and the main frame connection is transparent to the user.

In this paper we present examples of how to design and evaluate queueing experiments using RESQME. In section 2 we illustrate phase 1 of the modeling process: the graphical construction and editing of models. In section 3 we examine phase 2, model evaluation, and phase 3, output analysis. We present techniques for selecting model evaluation methods and for analyzing the results, and we do so through specific examples. Discussions involving the model evaluation and output analysis are supported by scenarios that illustrate the choice of procedures to obtain confidence intervals of desired accuracy, the determination of run lengths and the possible identification of the initial transient state from pilot runs.

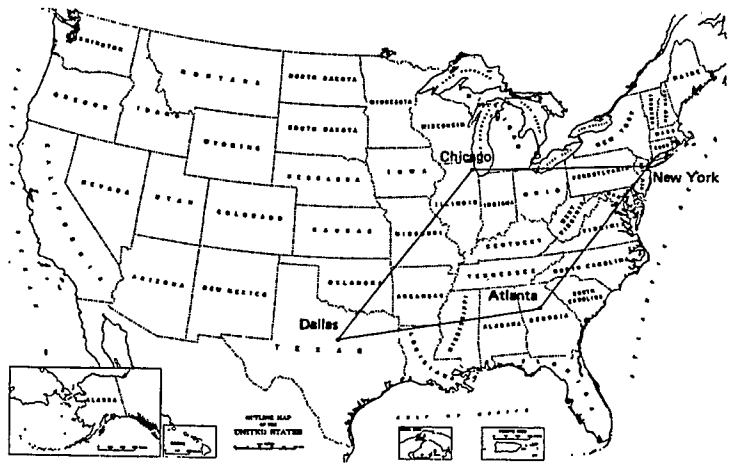

Figure 1. A Simple Four Node Communication Network.

2. MODEL SPECIFICATION

In this section we illustrate the use of RESQME for creating and editing a RESQ model definition and construct, as an example, a model of a simple computer communication network. The network to be modeled is shown in Figure 1 and consists of four nodes ("cities") connected by unidirectional communication links.

We begin building our model by defining the eight communication links; we will model each of these links as a single server, non-preemptive-priority queue. (As we will see, we will use priorities to model the fact that acknowledgment traffic is to be transmitted at a higher priority than standard data traffic.) To begin defining these queues, we first specify the model name and then select the "Create/Edit" option from the main RESQME command menu; at this point a pop-up Create/Edit submenu and a palette of RESQ icons appear. These menus and the palette appear on the graphics display, as shown in the lower portion of the screen in Figure 2.

In order to define a link, we select a queue icon from the palette using the graphics pointing device (e.g., a mouse or a joystick) and place the icon in the modeling area. When the palette icon is selected it is highlighted in red on the palette; once placed in the modeling area, the queue's color turns yellow to indicate that its attributes (e.g., its name, service time distribution, queueing discipline and priority structure) have not yet been specified. These attributes are specified in a context-sensitive form on the text screen either immediately after the queue is placed in the modeling area or by using the "Modify" option in the "Create/Edit" submenu at any later point in the modeling process. The text screen containing a completed form for the Chicago to New York link is shown in Figure 3.

Once the attributes of a modeling element (in this case, a queue) have been specified, they are immediately parsed for both syntactic and semantic correctness. Assuming that we have correctly specified the queue's attributes, its fill-color will be set to green. If we had made an error in the queue's attribute specification, error messages would have been displayed on the text screen and its fill-color would have be set to red. In addition to the eight active queues, we also define a single passive queue with four allocate nodes. One of each of these four allocate nodes will be used within each of the four "cities" to collect response time statistics.

Once we have defined the eight queues representing the eight inter-city communication links and the response time passive queue, we next specify the internal operation of each of the "cities". As our model is meant only for pedagogical purposes, the internal operation of each city will be quite simple; for more realistic models of this network, in which protocol mechanisms such as timeouts and flow control are considered, the interested reader is referred to Sauer and MacNair (1983).

In specifying the cities' operations, we will take advantage of RESQ's submodel facility. A submodel is essentially a parameterized template of an interconnected "subnetwork" of RESQ modeling elements. A submodel definition may be "invoked" several times (much the same way that a macro may be invoked several times in a programming language) to create multiple instances of that subnetwork. In our present model we will construct a single submodel for the operation of a generic "city" and then invoke this submodel four times in our main model, once for each "city" in the network. We note that submodels provide an important mechanism for constructing modular, well-structured and consistent performance models. Note, for example, that if at some point we need to modify the operation of the "city", we need only change the single submodel definition and these changes will then be automatically reflected in each of the cities' definitions.

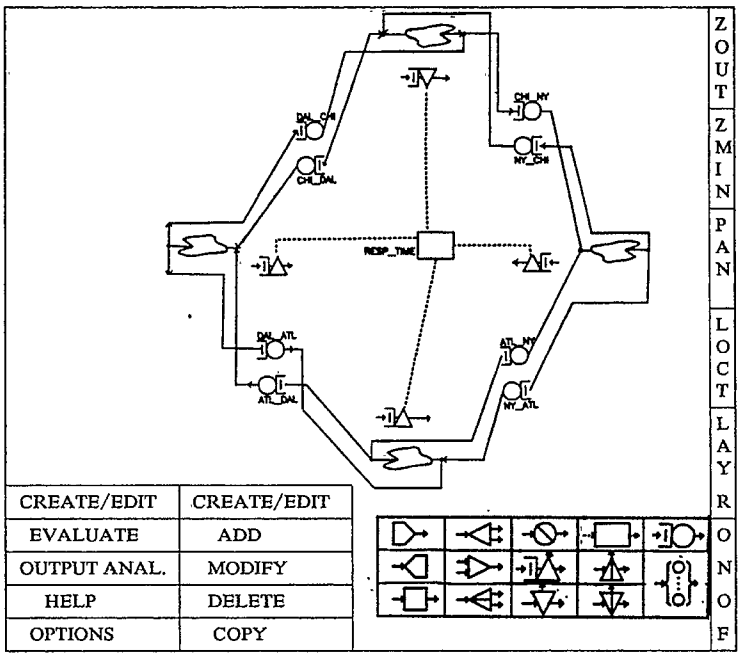

Figure 2. Main Model, Menus and Palette.

A submodel definition is created by selecting the "Layer" option in the menu at the far right of the modeling screen. This will cause a pop-up window to appear containing the names of all currently defined submodels as well as the options, "Up" and "New". Since we are defining the first (and only) submodel in this model, we select this last option. This causes a new modeling area to be created. This new modeling area is separate from the main modeling area and RESQME enforces this structure by allowing the modeler to move between these planes only through the explicit use of the "Layer" command. We may thus think of this as a new modeling "plane" or "layer", in which we may now proceed to define the elements of the city submodel.

As in the main model definition, we construct the submodel by picking icons from the palette, placing them in the modeling area, and defining their attributes in forms on the text display. However, before doing this, we first specify the submodel name and parameters by selecting the "Header" option from the "Create/Edit" submenu (second page of this menu reached by scrolling) and filling in the displayed form.

After specifying the submodel header and modeling elements, we next specify the routing of jobs between these elements. This is done simply by picking a set of "from" nodes (using the graphics pointing device), and then selecting a set of "to" nodes. The environment then automatically connects the "from" nodes to the "to" nodes using straight lines. As shown in Figure 4, one-to-one, one-to-many, or many-to-one routing may be drawn; many-to-many routing may also be specified. We may also draw a segmented line, stretch an existing line, or add articulation points to an existing line (without altering the routing definition itself) in order to create the desired pictorial representation of job routing.

In our submodel, a job (representing a data message) is generated at a source node, ENTRANCE, and then passes through the node parameter R__T; this node parameter will be matched with an allocate node in the previously-defined response time passive queue in the submodel invocation specification. A message visits this node in order to pick up a token to measure the time between its generation at ENTRANCE and the eventual receipt of its corresponding acknowledgment message at this city. The message then passes through the set node, MSG__VALS, where values are assigned to its job variables. Job variable 0 indicates the destination city code (a number drawn from a distribution at the set node); job variable 1 indicates the message length (again, a random number); job variable 2 indicates the city code for the city at which the message was generated (a parameter to the submodel), and job variable 3 indicates the message type (in this case, the numerical constant, DATA). Note that as we have not yet defined the constant, DATA, the set node definition contains an error. Thus, an error message will be displayed and the fill-color of the MSG__VALS set node icon will be set to red. This error will be automatically corrected when we later define DATA in the main model header.

Once a message passes through MSG__VALS, it arrives at the output node for this submodel. We will shortly connect this output node to two of the queues representing communication links in the main model and thus complete the routing of messages between cities. Messages arrive externally into the submodel via the submodel input node. One of three actions can then be taken on such an externally arriving message. First, if it is not destined for this node (as determined by the value of job variable 0), it is immediately

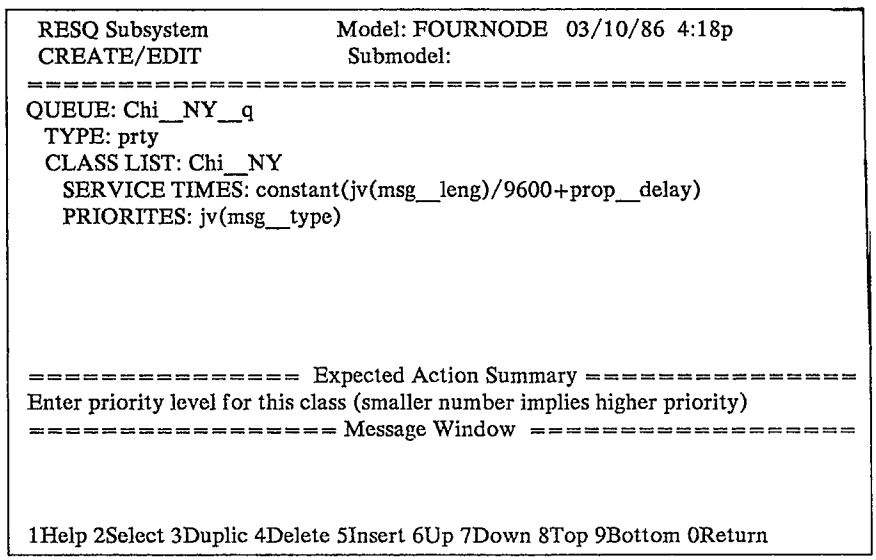

Figure 3. A Complete Attribute Definition.

routed to the output node. Otherwise if the message is a data message and destined for this city, it is routed to the set node, ACK__MSG, where it is transformed into an acknowledgment message by re-setting each of the job variables; the message is then routed to the output node. Otherwise, the message is known to be an acknowledgment message destined for this city, in which case it is simply routed to a sink node.

Once we have completed the submodel definition, we return to the main model by selecting the "Layer" command on the far right menu and selecting "Up" on the resulting pop-up submenu; the (still incomplete) main model definition is again displayed. We now complete the model definition by placing four invocation icons in the modeling area, defining their attributes (the corresponding submodel name and parameter values), and connecting the input/outputs of the queues representing the communication links to the outputs/inputs of the submodel invocations, as shown in Figure 2. The specific communication link used by a message emerging from the invocation output node is determined by the destination code carried by the message in job variable 0.

Finally, we recall that we have yet to define the header for the main model; this header contains the main model parameters, global definitions, and model size information. We thus select the Header option from the "Create/Edit" menu (as in the submodel definition) and complete the header definition form. As previously discussed, the error in the set node definition in our submodel is immediately corrected once we define the numerical constant DATA in the model header. As a final step, we supply the simulation control information by selecting the "method dependent" option from the menu and complete the associated form. At this point, the model has been completely and correctly specified and we may now enter the model evaluation phase in RESQME.

3. MODEL EVALUATION AND OUTPUT ANALYSIS

This section describes, by continuing the above example, suggested ways to approach the model evaluation phase and the output analysis phase. In the model evaluation phase, we assign values to parameters which have been defined in the model specification phase. We select "Evaluate" from the main command menu. A template, specific to that model, is displayed on the character screen, prompting us to enter the values for the necessary model parameters. We can specify the parameters for multiple runs of the model. We recommend that models be highly parameterized so that few changes are necessary when investigating several alternatives. In this example, the template consists of the parameters: mean message length and mean interarrival time. We enter, for example, the values 1400 bits and 0.2 seconds respectively.

When we complete the parameter specifications for each run, the model is then evaluated on the host by the RESQ solution component. While RESQ is evaluating the model on the host, we can modify the model by selecting the create/edit phase or examine output from another run in the output analysis phase. When the model evaluation is completed, the results can be graphically displayed on the PC.

The output analysis phase permits the display of the model's resulting performance measures directly along with the model diagram. We select "Output Analysis" from the main command menu. The output analysis phase submenu appears, as in Figure 5, providing the commands to retrieve the model data, to select the contents of the display, to specify the form of the output display and position of the graphs, to plot the results, and to remove the graphs. Once the model output is sent from the host, we can specify the form and content of the displays. We can place the resulting graphs wherever we wish on the display by the pointing device cursor using a rubberbanding rectangle; the resulting window locations and sizes are arbitrary. In the RESQME

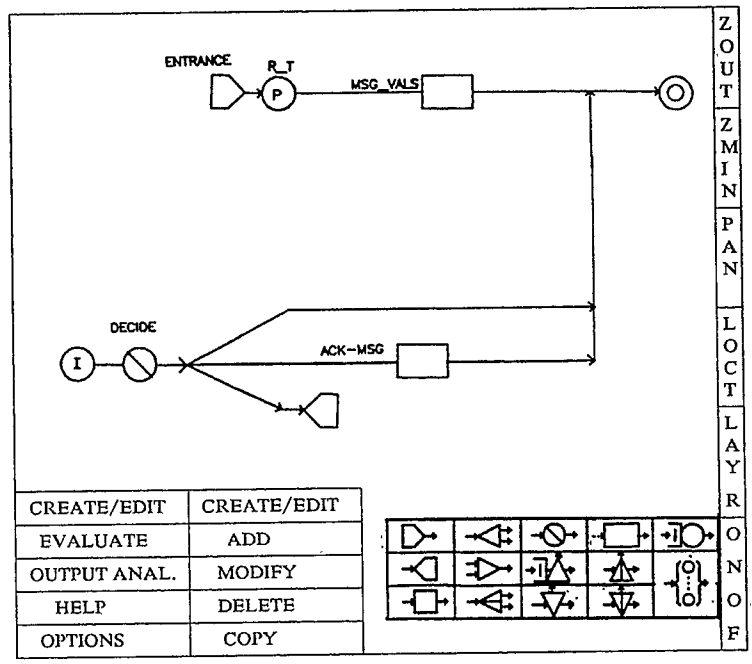

Figure 4. The "City" Submodel Definition.

design, the specification of the form of the graph is separated from the content, allowing the analyst to choose each independently. The most recent specifications of form and content are remembered by the system and become the default values for the next output chart. Form attributes can be specified to produce different types of graphs, to specify ranges of values, to select appropriate axis scales and labels, etc. The analyst selects the performance measures to plot and has the option to combine several performance measures in a single chart and to plot performance measures across runs and against other performance measures. RESQME also provides the analyst with the capability to perform calculations on the resulting performance measures, such as to project confidence interval widths for run length determination, to calculate moving averages, and to fit distributions.

Three confidence interval methods are available in the RESQ system: the regenerative method, the spectral method, and the independent replications method. No method will work well for all situations. In the absence of any prior information, we recommend that they be tried in the following order when performing a steady-state simulation: the regenerative method, the spectral method and lastly, independent replications. The reason for this recommended order is that when it is applicable, the regenerative method, which is a single run method, is normally more efficient in its use of CPU time than the other two methods. In addition, it does not require the analyst to identify the initial transient, and the initial condition can just be the regeneration state. Since the spectral method is also a single run method, we recommend its use if the regenerative method fails. The method of independent replications involves multiple runs and possibly discarding the initial transient from each run. The use of these methods will be discussed below. Other methods can be found in Law (1983), Law and Kelton (1982), Welch (1983). Sequential stopping procedures can be employed with most methods to provide an automatic mechanism for determining when confidence intervals have reached a specified level of accuracy. We will first apply the regenerative method to the above example.

The regenerative method (Crane and Lemoine 1977, Fishman 1978, Iglehart and Shedler 1980, Lavenberg and Slutze 1975, Law and Kelton 1982, Welch 1983) works on a single run of the simulation. It divides the run into independent blocks of data by identifying returns to a regeneration state. This method can only be used to analyze a model which reaches steady state. One major advantage that the regenerative method has over other methods is that the analyst does not have to be concerned with a choice for the initial condition. There also is no initial transient problem since the first regeneration cycle is no different from any of the others.

One of the most difficult problems related to using the regenerative method is the identification of an appropriate regeneration state. If the model which is being simulated contains only open chains, a good candidate for the regeneration state is the empty system. It is frequently the case that if the model does not return to the empty state sufficiently often for valid confidence intervals to be calculated, then no other regeneration state will be easy to identify. If the model is a closed model (a model with only closed chains), a good regeneration state is all the jobs located at a service center with a long exponential service time. Initialize all the customers at this service center.

Since our example is an open system, we use the empty

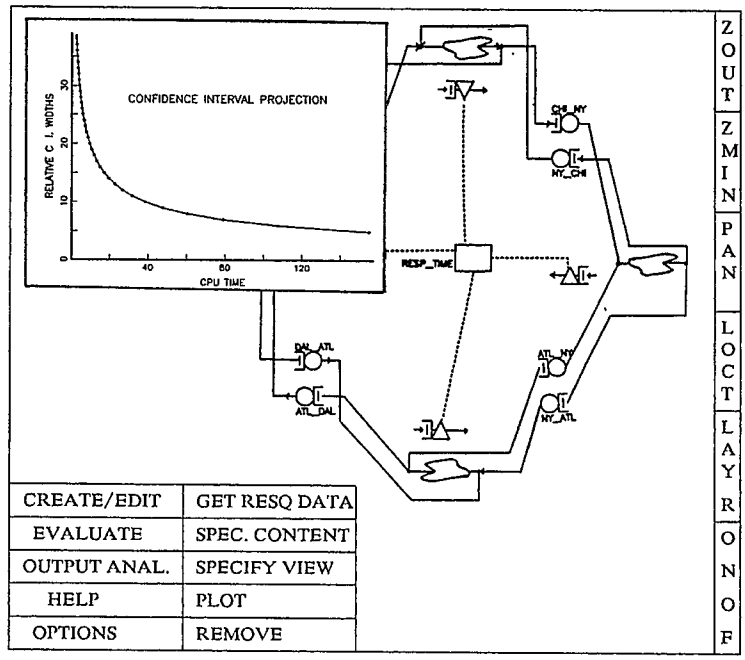

Figure 5. Estimate of CPU time versus Confidence Interval Width.

system as the regeneration state. To estimate how frequently the regeneration state will occur, we run the above model (with parameters: mean message length of 1400 bits and mean interarrival time of 0.2 seconds) for 200 departures from the response time queue. Since 200 departures occur in the middle of a regeneration cycle, RESQ continues to the end of the cycle which requires an additional 87 departures. During this short run, 2 regeneration cycles are produced. This is an indication that we should obtain a reasonably large number of cycles in a longer run. To make an estimate of a realistic run length, we increase the run to 500 departures. This longer run produces 6 regeneration cycles, a 38.4% relative confidence interval width for the mean queueing time, and required 2.63 CPU seconds for the solution. A graphical estimate of the required run lengths for different relative confidence interval widths is given in Figure 5. To produce this graph, we select "Specify View" in the output analysis submenu and position a rubberbanding rectangle to a desired area of the display. Then we select "Specify Content" and specify the y variable to be mean queueing time, the x variable to be cpu time, and indicate that a confidence interval width projection is to be produced. The resulting graph shows that approximately 155 seconds of CPU time is needed to produce a 5% relative confidence interval width.

We can now employ the sequential stopping procedure of RESQ (Lavenberg and Sauer 1977) to run until a 5% confidence interval width is detected automatically. We also choose to make 3 runs in the evaluate phase with mean message length 1400, 1500 and 1600 bits, respectively. The simulation program checks for the specified accuracy condition every 20,000 departures. We then plot the mean queuing time for the Response Time Queue verses mean service time for the three runs, producing the chart shown in Figure 6. The confidence interval widths are also shown on the plot. The number of regeneration cycles obtained for the above runs is 150, 101 and 41 at the specified stopping criterion.

The successful use of the regenerative method requires a sufficient number of regeneration cycles in a reasonable amount of computing time; we recommend that you observe at least 20 regeneration cycles and preferably at least 100. The regenerative method is based on the normal distribution which implies it is more accurate as the number of regeneration cycles increases. For short, fixed length runs, it is important to make certain that the last regeneration cycle in progress when the simulation stops is completed. This will reduce the bias in the results as discussed by Meketon and Heidelberger (1982). RESQ attempts to complete the last cycle. If it is not able to complete the last cycle, it will discard some events. This is an indication that the run should be continued until the last cycle is finished. If the pilot study produces no cycles or so few that an extremely long main experiment would be necessary, we recommend that the spectral method be tried.

The spectral method (Heidelberger and Welch 1981a and 1981b, Welch 1983) works on a single run of the simulation. However, it does not attempt to identify independent data as does the regenerative method or independent replications. Rather, it directly estimates and corrects for the effect of correlation which is found in the simulation output. It works only for models which exhibit equilibrium behavior and, as implemented in RESQ, produces confidence intervals only for the mean queueing times and points on the queueing time distributions. With this method, the analyst needs to be concerned with the initial transient phase.

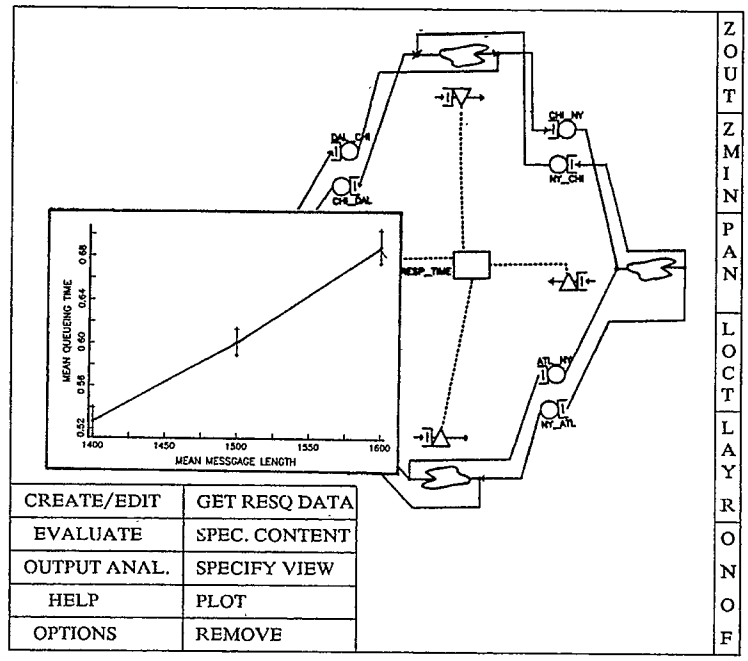

Figure 6. Performance Measures versus Mean Message Lengths.

In RESQ the spectral method will only produce confidence intervals for a steady-state simulation for the mean queueing time and points on the queueing time distribution. The method of independent replications would be necessary to perform a transient analysis or if the analyst needs confidence intervals for other performance measures. We complete the content specification information called for in the output analysis phase to calculate and display the following types of plots. For example, to determine the extent of the transient phase, we can produce a graph of individual queueing times averaged over all replications. A moving average calculation can be performed on this data to give an indication of the end of the transient phase. As was shown for the regenerative method, an estimate of the run length of a replication can also be plotted.

4. SUMMARY AND FUTURE DIRECTIONS

We have discussed the use of the RESQ modeling environment for model specification, evaluation and output analysis. We have emphasized the use of the model diagram as the focus of all phases of model development. A scenario has been presented to illustrate the use of this modeling environment to incorporate confidence interval methods and to estimate run lengths. The benefit of pilot runs to aid in the design of main experiments was also presented.

We have shown how the graphical input and output are an integral part of the modeling process. RESQME provides the necessary information about the structure of the queueing network and the resulting performance measures to solve the queueing problem and provides the information in an easy-to-use, easy-to-interpret graphical form. Furthermore, the user can directly specify and modify the graphics and the underlying attributes to evaluate alternatives. As a result, the analyst can access all the functionality of RESQ and control all aspects of the modeling process within this one graphics-oriented environment.

Our plans for extending this environment include tutorial facilities to guide the user in the model specification, confidence interval selection and output analysis phases. This facility would provide much assistance to new users in the form of a programmed learning environment. The ability to construct higher level icons to represent situations encountered in certain application environments would aid the model specification process. An include facility to draw on stored portions of models would make model construction more flexible. Animation of the model diagram would be helpful for debugging a model and for demonstrating its credibility to decision makers who are not necessarily familiar with queueing networks. A data base and work unit manager is necessary to coordinate data from various runs, various models and to keep a history of each model. The conversion of models constructed with the older version of RESQ to the graphical format will be useful to people using the older version. Models are frequently evaluated over a large parameter space in an attempt to approach an optimum solution. The new environment should provide some guidance in designing the parametric experiments and in employing some search techniques. The merging of the graphics with text in reports would be useful in preparing documents describing the model development.

ACKNOWLEDGMENTS

We would like to express our appreciation to Richard Gilbert for a number of fruitful discussions, which significantly influenced our choice of the design features of the environment. We are particularly appreciative of the contributions of Charles Sauer to RESQ. His insights into modeling, the design of RESQ and his simulation program have made RESQ the popular tool that it is. We are grateful to the many colleagues and RESQ users who have helped improve RESQ over the years.

REFERENCES

Browne, J. C., Neuse, D., Dutton, J. and Yu, K.-C. (1985). Graphical Programming for Simulation of Computer Systems. *Proceeding of the 18th Annual Simulation Symposium*, Tampa, FL, 109-126.

Chow, W.-M., MacNair, E. A. and Sauer, C. H. (1985). Analysis of Manufacturing Systems by the Research Queueing Package. *IBM Journal of Research and Development* **29**, 330-342.

Crane, M. A. and Lemoine, A. J. (1977). *An Introduction to the Regenerative Method for Simulation Analysis*, Lecture Notes in Control and Information Sciences, Vol.4, Springer-Verlag, New York.

Fishman, G. S. (1978). *Principles of Discrete Event Simulation*, Wiley, New York.

Gilbert, R. and Kleinoder, W. (1985). CNMGRAF - Graphical Presentation Services for Network Management. *Proc. 9th Data Communications Symposium*, Whistler Mountain, BC, 184-199.

Healy, K. J. (1985). Cinema Tutorial. *Proceedings of the 1985 Winter Simulation Conference*, San Francisco, 94-100.

Heidelberger, P. and Welch, P. D. (1981a) A Spectral Method for Confidence Interval Generation and Run Length Control in Simulations. *Comm. Assoc. Comput. Mach.* **24**, 233-245.

Heidelberger, P. and Welch, P. D. (1981b). Adaptive Spectral Methods for Simulation Output Analysis. *IBM J. of Research and Development* **25**, 860-876.

Iglehart, D. L. and Shedler, G. S. (1980). *Regenerative Simulation of Response Times in Networks of Queues*, Lecture Notes in Control and Information Sciences, Vol.26, Springer-Verlag, New York.

Kurose, J. F., Gordon, K. J., Gordon, R. F., MacNair, E. A. and Welch, P. D. (1986). A Graphics-Oriented Modeler's Workstation Environment for the RESearch Queueing Package (RESQ). Accepted for publication in the *ACM/IEEE Fall Joint Computer Conference.*

Lavenberg, S. S. and Sauer, C. H. (1977). Sequential Stopping Rules for the Regenerative Method of Simulation. *IBM J. of Research and Development* **21**, 545-558.

Lavenberg, S. S. and Slutz, D. R. (1975). Introduction to Regenerative Simulation. *IBM J. of Research and Development* **19**, 458-462.

Law, A. M. (1983). Statistical Analysis of Simulation Output Data. *Operations Research* **31**, 983-1029.

Law, A. M. and Kelton, W. D. (1982). *Simulation Modeling and Analysis*, McGraw-Hill, Inc.

MacNair, E. A. (1985). An Introduction to the Research Queueing Package. *Proceedings of the 1985 Winter Simulation Conference*, San Francisco, 257-262.

MacNair, E. A. and Sauer, C. H. (1985) *Elements of Practical Performance Modeling*, Prentice-Hall, Englewood Cliffs, N.J.

Meketon, M. S. and Heidelberger, P. (1982). A Renewal Theoretic Approach to Bias Reduction in Regenerative Simulations. *Management Science* **28**, 173-181.

Melamed, B. and Morris, R. J. T. (1985). Visual Simulation: The Performance Analysis Workstation. *IEEE Computer* **18**, 87-94.

Pegden, L. A., Miles, T. I. and Diaz, G. A. (1985). Graphical Interpretation of Output Illustrated by a SIMAN Manufacturing System Simulation. *Proceedings of the 1985 Winter Simulation Conference*, San Francisco, 244-251.

Sauer, C. H. and MacNair, E. A. (1982). The Research Queueing Package Version 2: Availability Notice. IBM Research Report RA-144, Yorktown Heights, New York.

Sauer, C. H. and MacNair, E. A. (1983). *Simulation of Computer Communication Systems*, Prentice-Hall, Englewood Cliffs, N.J.

Sauer, C. H., MacNair, E. A. and Kurose, J. F. (1982a). The Research Queueing Package Version 2: Introduction and Examples. IBM Research Report RA-138, Yorktown Heights, New York.

Sauer, C. H., MacNair, E. A. and Kurose, J. F. (1982b). The Research Queueing Package Version 2: CMS Users Guide. IBM Research Report RA-139, Yorktown Heights, New York.

Sauer, C. H., MacNair, E. A. and Kurose, J. F. (1982c). The Research Queueing Package Version 2: TSO Users Guide. IBM Research Report RA-140, Yorktown Heights, New York.

Sauer, C. H., MacNair, E. A. and Kurose, J. F. (1984). Queueing Network Simulations of Computer Communication. *IEEE Journal on Selected Areas in Communications* SAC-2, 203-219.

Sinclair, J. B., Doshi, K. A. and Madala, S. (1985). Computer Performance Evaluation with GIST: A Tool for Specifying Extended Queueing Network Models. *Proceedings of the 1985 Winter Simulation Conference*, San Francisco, 290-300.

Standridge, C. R., Vaughan, D. K. and Sale, M. L. (1985a). A Tutorial on TESS: The Extended Simulation System. *Proceedings of the 1985 Winter Simulation Conference*, San Francisco, 73-79.

Standridge, C. R., Vaughan, D. K. and Sale, M. L. (1985b). Presenting Simulation Results with TESS Graphics. *Proceedings of the 1985 Winter Simulation Conference*, San Francisco, 237-243.

Welch, P. D.(1983). The Statistical Analysis of Simulation Results. Chapter 6 in: *Computer Performance Modeling Handbook*, (S. S. Lavenberg ed.). Academic Press, New York.

AUTHORS' BIOGRAPHIES

ROBERT F. GORDON is a research staff member in the modeling and analysis software systems group at the IBM Thomas J. Watson Research Center. He received a B.S. in mathematics and physics from the City College of New York in 1964, an M.S. in mathematics from Carnegie Institute of Technology in 1965 and Ph.D. in mathematics from Carnegie-Mellon University in 1969. From 1968 to 1974, he was Manager of Mathematics and Programming for Hoffmann-La Roche, Inc., where he developed mathematical models for marketing, production planning and distribution. From 1974 to 1983, Dr. Gordon was Director of Information Management Services at Avis, where he headed the operations research, timesharing systems, and systems and programming groups. Dr. Gordon has been an adjunct professor at Hofstra University. He is a member of Phi Beta Kappa, Sigma Xi and ORSA.

Robert F. Gordon
IBM Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598
(914) 789-7170

EDWARD A. MACNAIR joined IBM in 1965. He has been on the research staff in the Computer Science Department at the IBM Thomas J. Watson Research Center since 1973. He is currently on the modeling and analysis software systems group developing modeling programs to solve extended queueing networks. In addition, he has been an adjunct staff member at the IBM Systems Research Institute, where he taught courses related to performance modeling. He is one of the developers of the Research Queueing Package (RESQ), a tool for the solution of generalized queueing networks. He is a coauthor with Charles H. Sauer of *Simulation of Computer Communication Systems*, Prentice-Hall, 1983 and *Elements of Practical Performance Modeling*, Prentice-Hall, 1985. He received a B.A. in mathematics from Hofstra University in 1965, and an M.S. in Operations Research from New York University in 1972. He is a member of ACM and ORSA.

Edward A. MacNair
IBM Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598
(914) 789-7561

PETER D. WELCH did his undergraduate work at the Universities of Wisconsin and Chicago. He received an M.S. in mathematics from the U. of Wisconsin in 1951. From 1951-56 he worked on radar signal processing at the Physical Science Laboratory of New Mexico State University. He received his M.S. in Physics from New Mexico State in 1956. He joined IBM Research in 1956 and has worked and published in the areas of speech recognition, queueing theory, pattern recognition, seismic signal processing, spectral estimation, Fourier analysis, system performance modeling, simulation output analysis and statistical graphics. He received his Ph.D. in Mathematical Statistics from Columbia University in 1963. He is an Adjunct Professor of Mathematical Statistics at Columbia and Area Editor of Operations Research for Simulation, Implementation and Validation of Stochastic Models.

Peter D. Welch
IBM Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598
(914) 789-7560

KURTISS J. GORDON received his B.S. in Physics from Antioch College in 1964, his M.A. and Ph.D. in Astronomy from the University of Michigan in 1966 and 1969, and his M.S.E.C.E. in Computer Systems from the University of Massachusetts in 1985. Until 1984, he taught in the Five-College Astronomy Department. Currently, he is a Senior Postdoctoral Research Associate in the Department of Computer and Information Science at the University of Massachusetts in Amherst. Dr. Gordon's interests include the display and interpretation of large bodies of data, modeling and performance evaluation, and graphical user interfaces. He is a member of the American Astronomical Society, Sigma Xi, ACM, and IEEE.

Kurtiss J. Gordon
Department of Computer and Information Science
University of Massachusetts
Amherst, MA 01003
(413) 545-4207

JAMES F. KUROSE received a BA degree in Physics from Wesleyan University in Middletown, Conn. in 1978 and an MS and PhD degree in Computer Science from Columbia University in 1980 and 1984, respectively. Since 1984, he has been an Assistant Professor in the Department of Computer and Information Science at the University of Massachusetts, Amherst, MA., where he currently leads several research efforts in the areas of computer communication networks, distributed systems, and modeling and performance evaluation. He has also been associated with the performance modeling methodology group at the IBM T.J. Watson Research Center as a consultant since 1980 and has served as a consultant for various other companies as well. Professor Kurose is a member of Phi Beta Kappa, Sigma Xi, IEEE, and ACM and the IEEE Technical Committees on Computer Communications, Distributed Systems, and Computer-Aided Modeling of Communication Systems.

James F. Kurose
Department of Computer and Information Science
University of Massachusetts
Amherst, MA 01003
(413) 545-1585