2-25-1988

Higher Level Modeling in RESQME

Robert F. Gordon Ph.D.
Molloy College, rfgordon@molloy.edu

Kurtiss J. Gordon

Recommended Citation

Research Report

Higher Level Modeling in RESQME

Robert F. Gordon, Edward A. MacNair

IBM Research Division
T. J. Watson Research Center
Yorktown Heights, N.Y. 10598

Kurtiss J. Gordon, James F. Kurose

Department of Computer and Information Science
University of Massachusetts
Amherst, Mass. 01003

HIGHER LEVEL MODELING IN RESQME

Robert F. Gordon and Edward A. MacNair
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 REScarch Queueing Package Modeling Environment (RESQME) is a graphical workstation environment for iteratively constructing, running and analyzing models of resource contention systems. It is built on top of the REScarch Queueing Package (RESQ) which provides the functionality to evaluate extended queueing networks. In this paper we describe the high-level building component design for RESQME. The modeler is provided with tools to create his own icons and to associate them with submodels. He then uses these building blocks to construct his model. This capability extends the fundamental building blocks of RESQ and allows the user to create models with objects directly related to his application domain.

INTRODUCTION

The RESearch Queueing Package Modeling Environment (RESQME) (Kurose et al. 1986, Gordon et al. 1986 and Gordon et al. 1987) is a graphical workstation environment for iteratively constructing, running and analyzing models of resource contention systems. It is built on top of the REScarch Queueing Package (RESQ) (Chow, MacNair and Sauer 1985, MacNair 1985, MacNair and Sauer 1985, Sauer and MacNair 1982, Sauer, MacNair and Kurose 1982a, 1982b, and 1982c) which provides the functionality to evaluate extended queueing networks, analytically or by simulation.

Queueing problems arise whenever jobs have to wait for service from limited resources, for example, units on an assembly line, messages in a communications network, or transactions in a computer system. Software packages provide capabilities for the construction of models of the queueing system and for the evaluation of its performance under alternative conditions.

In analyzing a queueing problem, typically the modeler first creates a graphical representation of the system to be studied. With early packages like GPSS, SIMULA and SIMSCRIPT, he would then translate that diagram into the syntax of the package. That textual representation is the basis for his interaction with the computer program. He goes through a verification and modification cycle of iteratively examining the results by translating them to his domain and then translating the changes back to the textual model by editing the model and rerunning the package.

Much of this translation effort of the modeler can be eliminated by allowing the modeler to use the network diagram directly as the means to specify and modify the model to the computer and view the results from the computer. References of graphical systems include Browne et al. 1985, Melamed and Morris 1985, Sinclair, Doshi and Madala 1985, and Standridge, Vaughan and Sale 1985.

The graphical environments move the interface line where the modeler interacts with the computer from the textual representation to the graphical description. This is accomplished in RESQME both for the input specification and the output analysis, making it easier for the modeler to specify his model and interpret the results. The modeler can think in terms of the graphical representation of the model and need no longer be concerned with translating his vision of the problem into the syntax requirements of the textual representation and translating the computer output from tabular to graphic form. Iterative changes and output requests are accomplished by manipulating the graphics representation rather than the textual.

With the higher-level modeling component of RESQME, we further move this interface line closer to the modeler's domain, removing the need to translate from his domain to the graphical elements of the language. We allow him to create and specify models with his own graphical constructs.

In this paper, we discuss the higher-level building component of RESQME. The overall design of RESQME is outlined in the next section. The tools and meth-

Figure 1. Main Menu
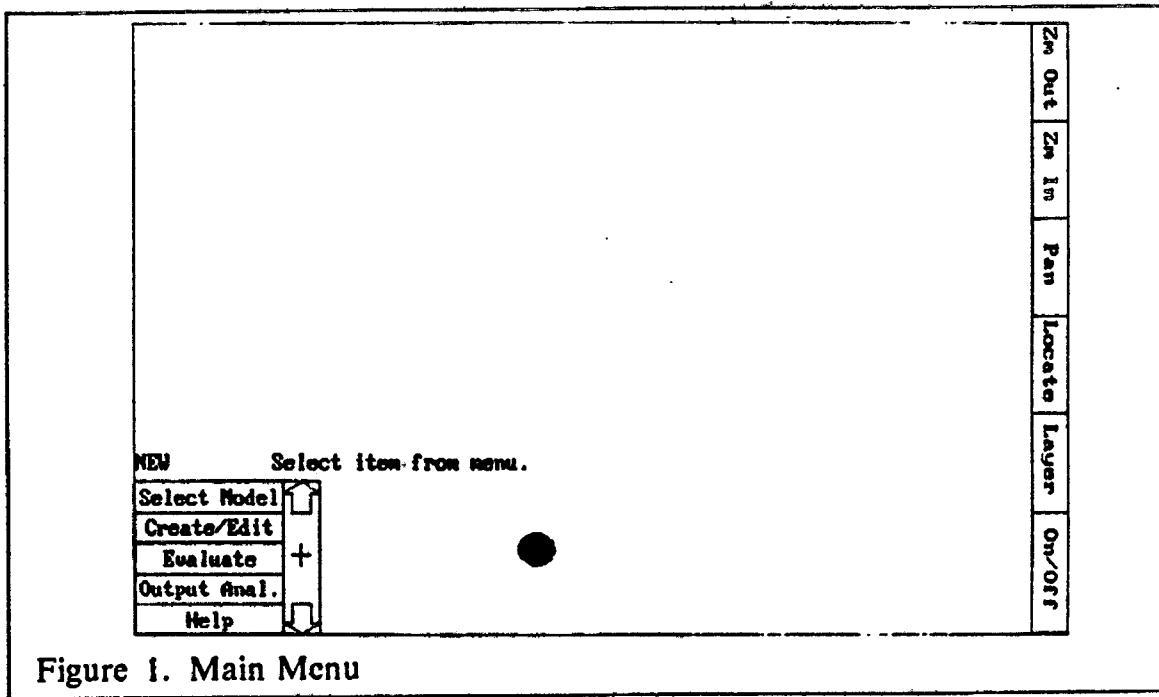
odology to construct and use the higher-level elements are described in the third section. In the final section, we present an example to demonstrate the use of higher-level building blocks.

DESIGN OVERVIEW

Here we introduce the RESQME environment by presenting an overview of the user interaction.

RESQME is a menu-driven interface with direct manipulation of the objects that comprise the network and its output charts. The network diagram serves as the interface between the modeler and the computer. The objects of the network diagram are icons which represent the fundamental RESQ elements (such as a service center, source of jobs, sink) and the links which connect the elements to form the job routing. The modeler is also provided with tools to extend the fundamental elements and routing by combining them into submodels and creating a new icon to represent the submodel. The modeler can then link these higher-level elements to other higher-level elements and to the basic building blocks to construct his model.

RESQME runs on a workstation consisting of a PC with a graphics display and optional pointing device. The PC is connected to a host computer. All user interaction, graphic display and specification is on the PC, while the computation-intensive evaluation of the model takes place on the host. The co-operative processing environment is transparent to the user. The uploading of files and commands and downloading of results is done automatically by RESQME.

The modeling process is organized into three tasks in the RESQME environment. The first task is model formulation and modification: the Create/Edit Task. In this task, the queueing model is constructed and edited. The second task is model solution: the Evaluate Task. The modeler can assign parameter values and execute the solution component in this task. The third task is called the Output Analysis Task. In this task, the modeler views the results of the evaluation. The control of the three tasks is handled by the Main Menu shown in Figure 1 on page 2. The modeling area is shown above the menu. Selecting a task in the Main Menu causes the approprate Task Menu to appear for selection. The tasks can be cycled and repeated in any order until the modeler completes his analysis.

To build the model, the modeler selects the Create/Edit Task. He selects icons from the icon palettes and places them on the modeling plane. The icon palettes contain both the fundamental icons and user created higher-level icons. When either type of icon is selected and placed on the modeling surface, a template appears for the modeler to supply underlying information.

Information is provided to the computer and to the modeler on two levels. One level is the graphics (network diagram, its components and the output charts). A second level of information consists of the underlying attribute data for the graphic objects. Thus a queue icon has attribute data consisting of its name, type of queue, queueing discipline, service time, etc. Similarly a chart has attribute data consisting of its type (line, bar, histogram), color, contents. The network diagram has attribute information consisting of the model name, solution method, parameter names, run control limits. The underlying attribute data appear on a character template which physically can be either a separate display or a mapping of the template onto the one graphics display.
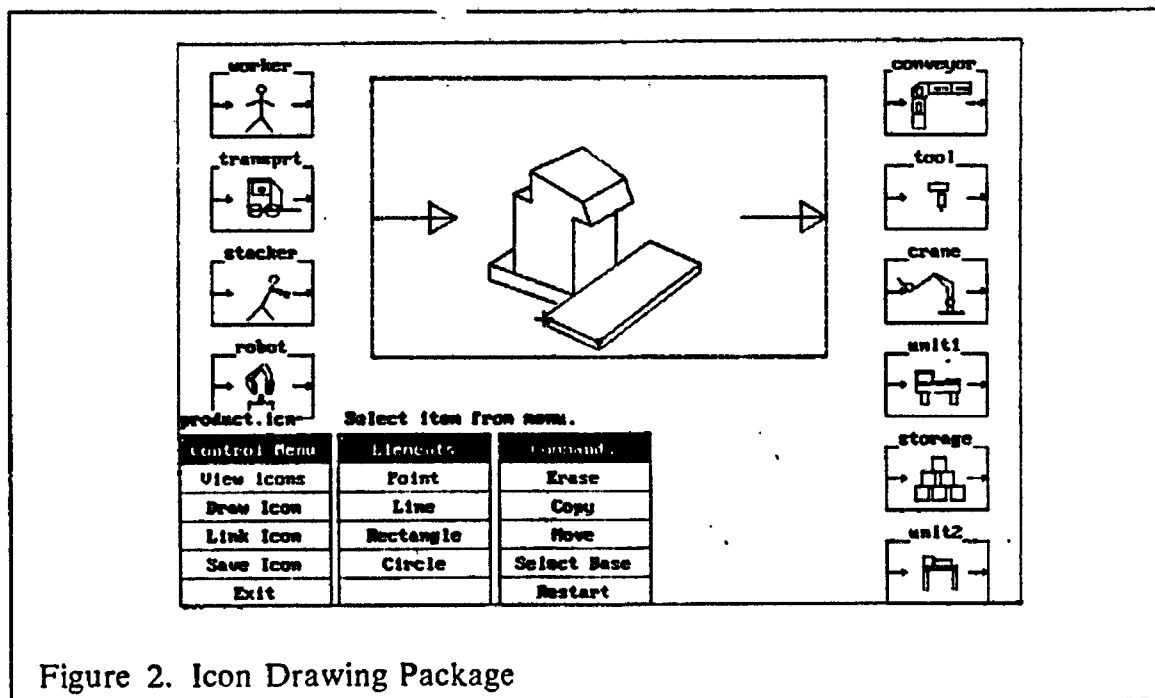

Figure 2. Icon Drawing Package

In particular, when a user-created icon is selected and placed on the modeling surface, a corresponding submodel template appears, allowing the modeler to provide the textual attributes. The information provided to the modeler is the submodel type that this icon represents (identified by the submodel name) and the prompts (parameter names) for entering the required parameters (if any) for this submodel. This information can be filled in at this point or at any other time in the modeling process. (If the submodel has not yet be created, the parameter prompts are blank.)

The user-created icon can be directly connected in the model network to other user-created icons, as well as to the fundamental RESQ elements. Icons are connected by choosing the job routing icon and pointing to the icons to be linked in the desired order. N-to-1 and 1-to-N routing capability is provided. The attribute information for routing allows the modeler to supply probabilities and/or conditional predicates for any of the routing links.

The modeler can view and modify the higher-level model, as well as the corresponding details of the lower-level submodels. The submodels used in the model can each be viewed by layering through the different levels of the model. A tree structure is formed with the main model at the root and the submodels forming the branches. To view or modify a submodel represented by a higher-level icon, the modeler selects the "Detail" menu item and points to the higher-level object that he wants to see. The submodel that that icon represents is then displayed on the modeling plane and can be edited in the same way as the main model. In addition, the modeler can traverse the levels of the model's hierarchy by selecting the "Layer" menu item. That selection displays a choice of the name of the (sub)model above the current layer and the submodels at the layer below. Choosing one of these allows the modeler to traverse the levels of the model. Included in the choice is the library of submodels. So a modeler can select pre-packaged submodels to include in his model as well.

In the Evaluate Task, the modeler then provides the values for the model parameters and selects execute to upload the model to the host and to run the simulation. While it is running on the host, the modeler is free to modify or view this model or any other models. He is signaled when the run is complete.

Selecting Output Analysis, the modeler then downloads the results from the host to the workstation, where he can select the output measure to plot. This is done by pointing to the object on the diagram for which information is desired. A list of that object's performance measures is provided. The modeler chooses any number of them and selects plot to view the output results on the model diagram.

TOOLS FOR HIGHER-LEVEL MODELING

In this section, we describe the Icon Drawing package that provides the tools for the user to construct his own icons and to link them to submodels and the access to these user-created icons in RESQME.
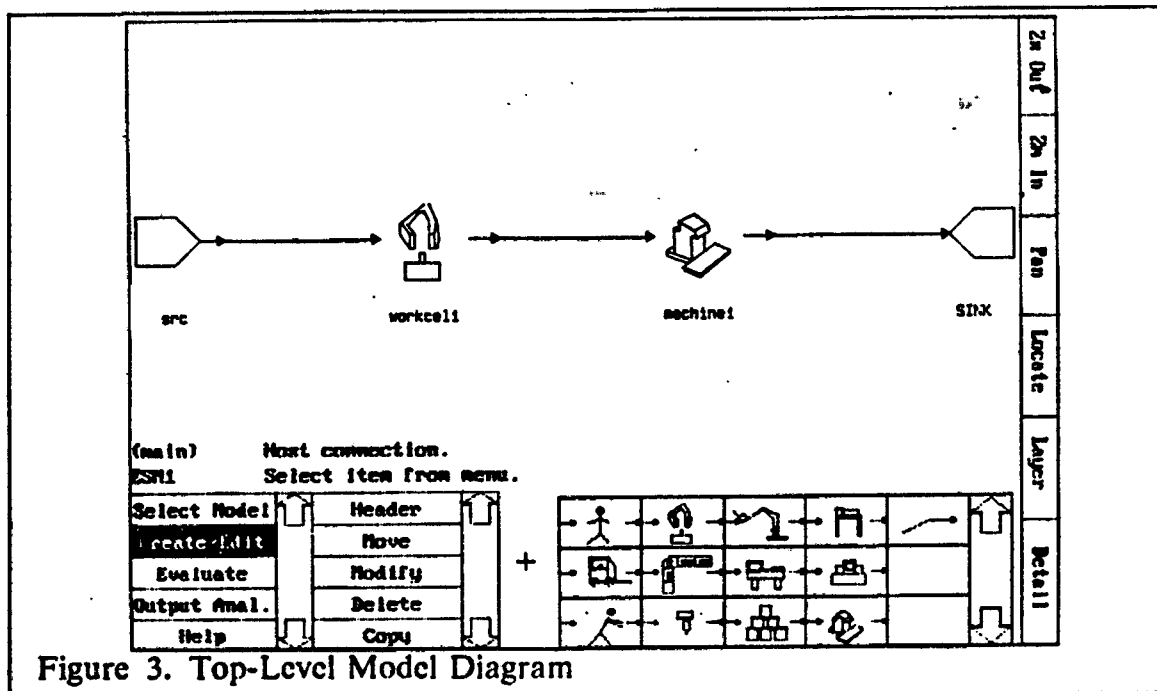
Figure 3. Top-Level Model Diagram

Running the Icon Drawing package produces a display as shown in Figure 2 on page 3. The Control Menu provides the modeler with the ability to view existing user icons, draw new ones, link them to submodels by name, and to save the resulting object.

User-created icons are saved in files. Each file contains an arbitrary number of icon palettes usually related to a particular application area. By providing the Icon Drawing package the name of such a file, the modeler can view the icons for that application area. He can then draw additional ones either from start or by using an existing icon as the base. The icons are simple drawings consisting of line, rectangle, and circle elements. Editing functions are provided to erase, move and copy these basic drawing elements. A submodel name is provided to the icon which links it to a submodel type. The submodel type can exist or can be created after the fact. Saving the icon, enters its drawing and linkage on the icon file. The resulting icon structure defines a new object to RESQME, consisting of its drawing elements and pointers to its associated submodel structure.

In RESQME, all the fundamental RESQ objects comprise two icon palettes. Any user-created icons form additional palettes. Selecting a model in RESQME causes all the user-created submodel icons associated with the model to be brought into memory.

The model diagram contains fundamental icons as well as any user-created icons, and the icon palette contains all the user-created icons associated with the application area. The modeler can use them to create and modify the model at any level. As with the fundamental icons, the user-created icons are selectable from the icon palette, positioned on the modeling area, linked in the network, panned,

rotated and zoomed. Essentially the high-level objects have all the functions as the built-in fundamental objects.

Additional icon palettes from other applications can be brought into RESQME by switching palettes in the menu.

MANUFACTURING EXAMPLE

As a simple example to demonstrate the use of higher-level building blocks, we present a model of a portion of a manufacturing line containing a robotic workcell and a simple machine with failures. Figure 3 on page 5 illustrates the highest level of the model with two user-created icons, one for the robotic workcell and one for the machine. Jobs enter the model at a source called SRC, and then flow serially through the workcell (WORKCEL1) the machine (MACHINE1) and leave at the SINK.

If we layer down to the submodel which represents the workcell, we see the portion of the model shown in Figure 4 on page 7. This submodel contains only fundamental RESQ objects. There is a passive queue for an input staging area, one for the robot and a third one for the output staging area. The jobs wait in a buffer until the input staging area is free. When it is free, the first job begins an orientation step. After the orientation is completed, if the robot is available, the robot picks up the job and moves it to the processing station and waits with the job until the processing is complete. The robot then moves the job to the output staging area and puts it down if it is free. If the output staging area is still occupied, the robot waits with the job. After releasing the job, the robot is free to return to the input staging area to pickup the next job that has been oriented. This submodel contains parameters for the various processing times used in the submodel. This submodel is based on a model described by Medeiros and Sadowski 1983.

If we layer back up to the model level and down to the machine submodel we see the diagram shown in Figure 5 on page 8. The flow through this submodel from the higher level is through the icon labeled PROCESS. The other path shown through nodes labeled DOWN and UP represents tool failures. When the job in the lower path is at UP, the machine is working properly. When the failure job is at DOWN, the machine has failed and is being repaired. This submodel contains parameters for specifying the processing time, time to failure and repair time distributions.

If we refer back to Figure 3 on page 5, we can see how easy it is to construct models when a library contains submodels representing the type of processing we are interested in modeling. The two user-created icons in this model refer to submodels which are available for use with the RESQME package. When the modeler places one of the user-created icons on the modeling area, a template appears prompting him to supply the values for the submodel parameters. More complicated models can be constructed by interspersing the use of submodels from the library with the fundamental RESQ building blocks.
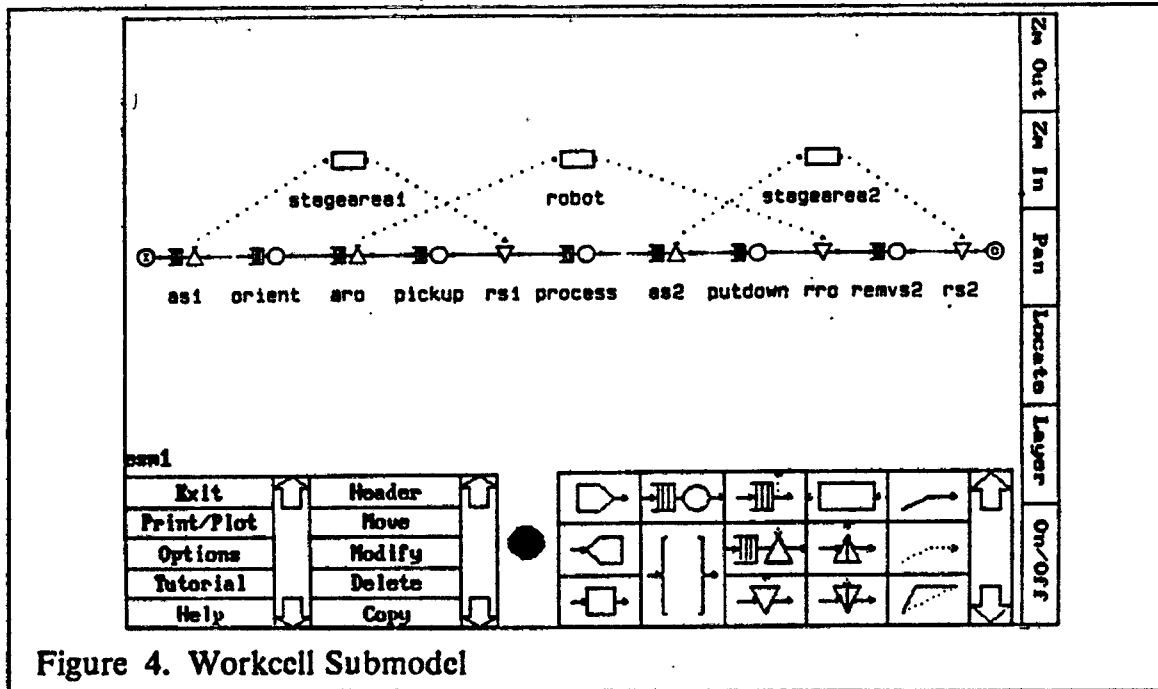
Figure 4. Workcell Submodel

Figure 6 on page 9 displays a chart of performance measures produced on top of the workcell submodel showing the cycle time distribution for the robot with upper and lower confidence interval data.

CONCLUSION

The higher-level modeling component allows the modeler to work with objects directly related to his application domain. Libraries of these objects provide the means to share and reuse submodels. Because the high-level building blocks are representations of RESQ submodels and because the RESQ elements themselves are also available, this design has unique flexibility and power.

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.

Gordon, R. F., MacNair, E. A., Welch, P. D., Gordon, K. J. and Kurose, J. F. (1986). Examples of Using the REScarch Queueing Package Modeling Environment (RESQME). *Proceedings of the 1986 Winter Simulation Conference*, Washington, D.C., 494-503.
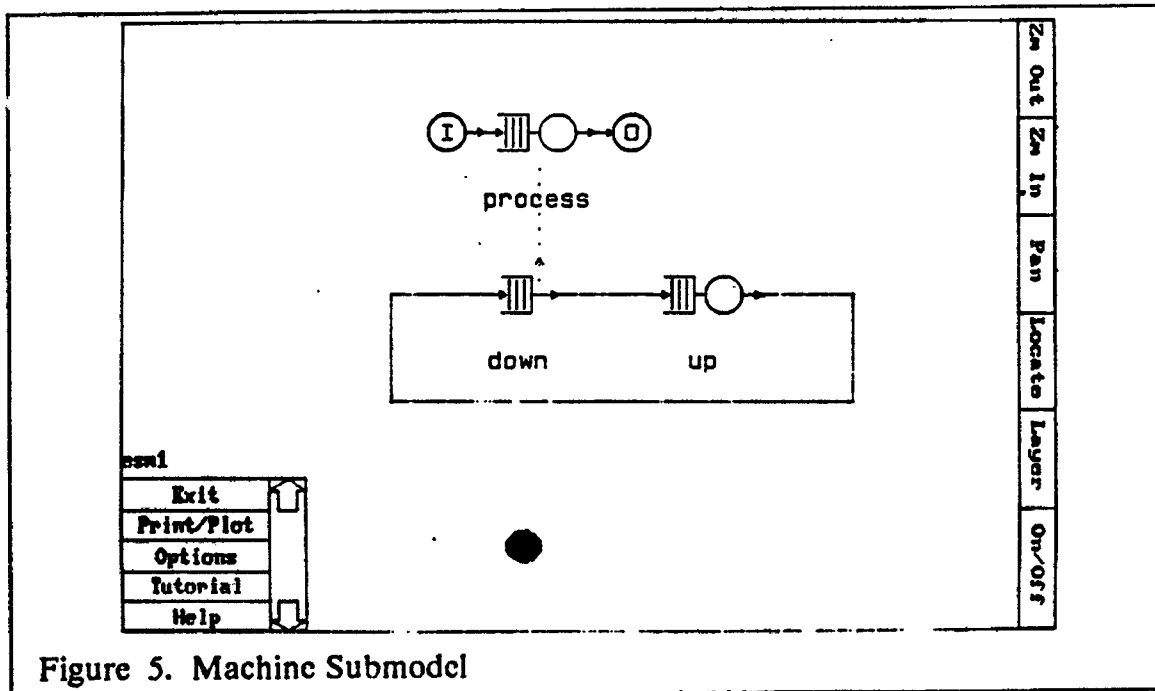
Figure 5. Machine Submodel

Gordon, R. F., MacNair, E. A., Gordon, K. J. and Kurose, J. F. (1987). A Visual Programming Approach To Manufacturing Modeling. *Proceedings of the 1987 Winter Simulation Conference*, Atlanta, December, 465-471.

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). *1986 Proceedings Fall Joint Computer Conferei:⁻ᵒ*, Dallas, 719-728.

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.

Medeiros, D. J. and Sadowski, R. P. (1983) Simulation of Robotic Manufacturing Cells: A Modular Approach. *Simulation* **40**, 3-12.

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

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.
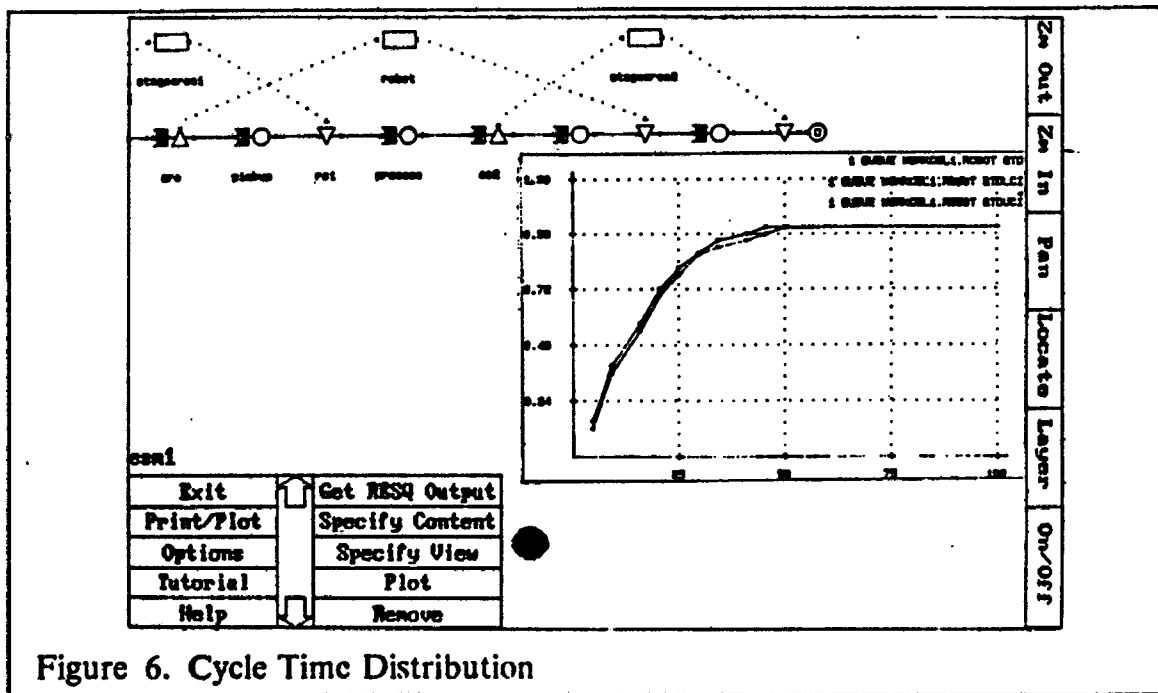
Figure 6. Cycle Time Distribution

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.

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. (1985). A Tutorial on TESS: The Extended Simulation System. *Proceedings of the 1985 Winter Simulation Conference*, San Francisco, 73-79.