Proceedings of the 1987 Winter Simulation Conference
A. Thesen, H. Grant, W. David Kelton (eds.)

A VISUAL PROGRAMMING APPROACH TO MANUFACTURING MODELING

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 RESearch Queueing Package Modeling Environment (RESQME) is an interactive, graphics-oriented workstation environment for iteratively constructing, running and analyzing models of resource contention systems. It is built on top of the RESearch Queueing Package (RESQ) which provides the functionality to evaluate extended queueing networks. In this paper, we describe how to create a "pull" system, a manufacturing line with finite buffers, using the graphical interface and hierarchical modeling capability. This serves as an example of the kind of manufacturing submodels that can be created in this system and then used for higher-level modeling.

1. INTRODUCTION

A key element in the efficient design and manufacture of superior products is the ability to model existing and proposed manufacturing systems and to easily and accurately estimate the effects of alternative manufacturing procedures. Simulation is the method often used to analyze the performance of manufacturing systems, allowing the analyst to take into account contention for machine time, complex routing decisions, and specialized equipment.

The modeling and evaluation process is iterative and graphical in nature (Browne et al. 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 engineer/planner often first abstracts the important elements of the manufacturing system by constructing a diagram of the manufacturing system, then converts this view of the system into the input form for the simulation package, runs the package, and then relates the output of the package back to the real system, often converting it to graphics for easier interpretation. The planner then modifies the system model and compares its resulting performance measures. He cycles through this experimental procedure until satisfied with the results. It is important therefore to provide the engineer/planner with one graphical environment that supports the iterative and visual nature of modeling and evaluation. The graphical environment allows the technical people closest to the process and with the best technical understanding of the problems to be the ones that construct and analyze the models without being forced to translate their view of the system into the syntax of a programming language.

The RESearch Queueing Package Modeling Environment (RESQME) (Kurose et al. 1986 and Gordon et al. 1986) is an interactive, graphics-oriented workstation environment for iteratively constructing, running and analyzing models of resource contention systems. It is built on top of the RESearch Queueing Package (RESQ) (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) which provides the functionality to evaluate extended queueing networks. RESQME runs on a workstation consisting of a PC with a graphics and a character screen display. The PC is connected to a host computer so that the computation-intensive evaluation is done on the host while the graphic specification and display is done on the PC.

We describe the fundamental modeling elements of RESQ and their graphical use in RESQME by building a simple model of a manufacturing line in section 2. In section 3 we extend this example to create a manufacturing line with finite buffers, first to model a simplified "pull" system and then a true pull system, a continuous flow manufacturing system. In a continuous flow manufacturing system, a job can only proceed to the next machine when there is room at that machine to receive it. Each machine therefore has a finite input buffer that blocks entry when full. In section 4 we describe the use of submodels to represent this process as an example of the higher-level modeling facility of RESQME. We discuss the design of this higher-level modeling facility in section 5.

2. RESQ MODELING ELEMENTS

We describe in this section the fundamental modeling elements in the RESQ language used in this paper. We will demonstrate their use in RESQME by constructing a simple manufacturing line with three machines.

The fundamental RESQ elements are the jobs that travel through the system, service centers that provide service to the jobs, passive resources that provide storage areas for jobs, and chains that provide the routing paths for the jobs. Jobs can enter the network according to an arrival pattern from a "source node" or be initialized in the network. Jobs travel through the system by moving from one node to the next in the network according to the route specified. Each job has a set of variables associated with it that defines its attributes. Attributes might be the type of job, its service time, the nodes it visited. These attributes can be used to determine routing for a specific job or work demand at a server, etc.

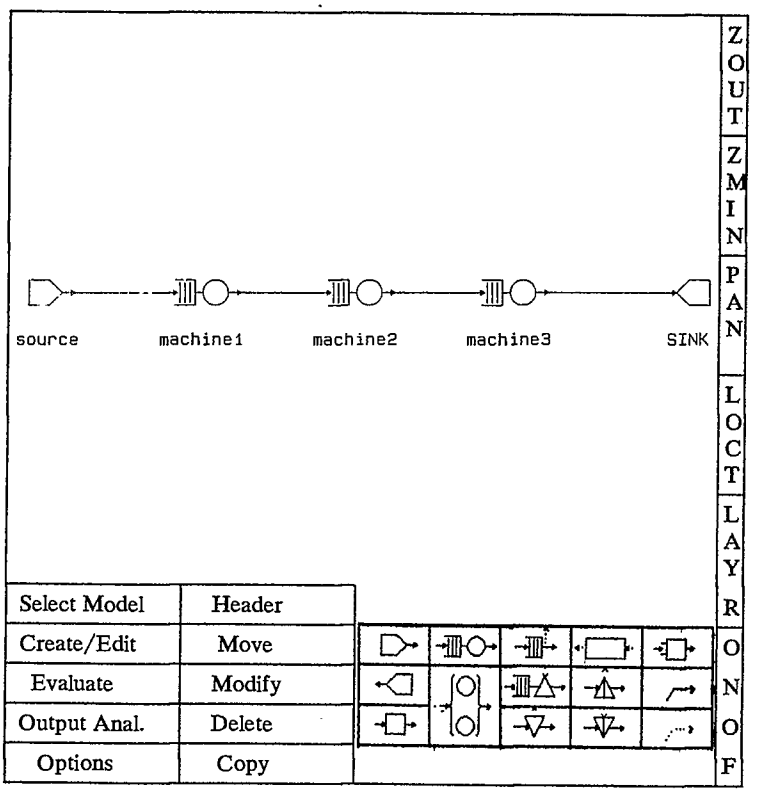

Figure 1: RESQME Graphics Screen

A service center contains one or more servers, one or more waiting lines and a scheduling algorithm to select jobs for service. A job is routed to the appropriate waiting line for service, and the waiting line is thus a node in the network. A machine can be represented by the service center icon by specifying the number of servers, the service distribution, the number of waiting lines and the scheduling algorithm. Names are given to the waiting lines and to every other node in the network to identify the routing path through the network.

A node called the "sink" is used to remove jobs from the network.

With this overview of the building blocks of RESQ, we will now show how to construct a simple manufacturing line with three machines using RESQME. The manufacturing line we will construct here is a "push" system, i.e., as a job is completed on one machine it is sent to the next in the line, and it is assumed that each machine has an infinite buffer to accept jobs. This policy causes a higher work in process level than the pull systems we will model in section 3.

Figure 1 shows the RESQME graphics screen display with the view into the modeling area on the top 3/4 of the screen, command menus and the icon palette on the bottom 1/4 of the screen, and a vertical screen management menu on the right border of the screen. We will be using four of the icons to build the manufacturing line. The icons corresponding to the source and sink nodes are shown in Figure 1 in the first row, first column and the second row, first column, respectively, of the icon palette. The icon for the service center is in the first row, second column. The icon used for routing jobs is in the second row, fifth column.

The modeling process is organized in three phases in RESQME, represented by the menu items Create/Edit, Evaluate, and Output Analysis. The Create/Edit phase allows the modeler to specify the model by building or modifying a diagram of the network and its underlying attributes. The Evaluate phase allows the modeler to provide values for the parameters and run the model on the host. The Output Analysis phase allows the modeler to view the resulting performance measures. The modeler can iteratively go through these phases until satisfied with the results. Menu selections and drawing take place on the graphics screen by using a pointing device such as a mouse or joystick. Filling in templates to identify the attributes associated with the graphics objects takes place on the character screen using the keyboard.

When we select the Create/Edit phase menu item, say with a mouse, the submenu that provides the commands to build and modify the model is displayed along with the icon palette as shown in Figure 1. We specify in the model header template on the character screen that the desired solution technique should be simulation (the modeler can also select the analytical solution technique), and we give names to the parameters to be used in the model. We name four parameters for this example: tba for time between arrivals, st1, st2 and st3

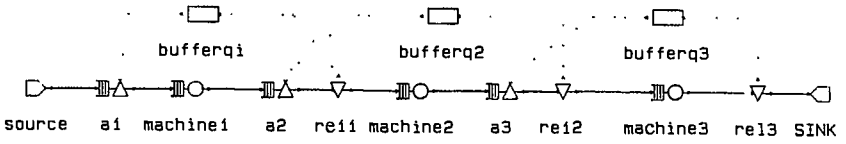

Figure 2: Diagram of Simplified Pull System

for mean service time at machines 1 to 3, respectively. We then begin drawing the manufacturing line by pointing to the source node and placing it on the modeling screen. We are prompted to provide the arrival distribution on the character screen template. We enter "tba". We will provide a value for tba during the Evaluate phase; the default is that that value will be used as the mean of an exponential distribution. We could specify any distribution and parameters. We select the icon representing a service center and place it on the modeling screen. In the corresponding template for this icon, we use the defaults to specify that there is one server, one waiting line and first-come-first-served service algorithm. We enter the parameter name st1 for the mean service time for this machine 1. We place the remaining two machines on the modeling surface in the same way (or by copying the existing one) and supply the appropriate parameter for mean service time. We then point to the sink icon and place it on the modeling screen.

To route the manufacturing jobs, we select the routing icon, and point to the nodes on the modeling screen in the sequence to be visited: source, machine1, machine2, machine3 and sink.

This essentially completes the model of a push system. From the workstation environment we can specify run control information, provide parameter values and execute the model on the host, and view the results graphically on the PC. In section 3, we describe how to modify this model to represent a a pull system.

3. MODEL WITH FINITE BUFFERS

Starting with the model of the manufacturing line that we developed in section 2, we will create a simplified pull system corresponding to the system described in Chow, MacNair and Sauer (1985) and then model the type of pull system we defined as a continuous flow manufacturing system.

We introduce one new building block, the passive queue, that we will use in this example to represent the total number of jobs that can be kept at a machine: the finite input buffer plus 1 for the job in service at that machine. We use the parameter names buffer_n, with n the machine number 1, 2, or 3, to specify the buffer size for each machine. There are two nodes that are used in conjunction with the passive queue: the allocate node and the release node. When a job reaches an allocate node, it tries to get space from the passive queue. We call each space unit a token. If there are tokens available, the job will get the required tokens (in this case 1) and move to the next node in its route; otherwise it will wait until the needed tokens are available. When a job reaches a release node it gives back its tokens to the passive queue. The icon for the passive queue is in the first row, fourth column of the icon palette in figure 1; the icon for the allocate node is in the second row, third column; the icon for the release node is in the third row, third column. Token flow from the passive queue to its allocate node and from a release node back to the passive queue is represented by a dotted line which is drawn with the icon in the third row, fifth column of the icon palette.

Figure 2 shows the resulting diagram of the simplified pull system model. Note that there is a passive queue for each machine that prevents a job from entering that machine until there is room for it. The passive queues overlap, because of the fact that a job cannot release its spot at its current machine until it is allocated a spot on the next machine down the line.

This model does not strictly represent a continuous flow manufacturing system. A job that completes processing at a machine will free the machine in this model and effectively switch places with a job in the buffer until it can proceed to the next machine. Thus the buffer serves as both an input and an output buffer, and additional jobs can be processed while an earlier job waits to enter the next machine. By incorporating another passive queue at each machine, we can prevent

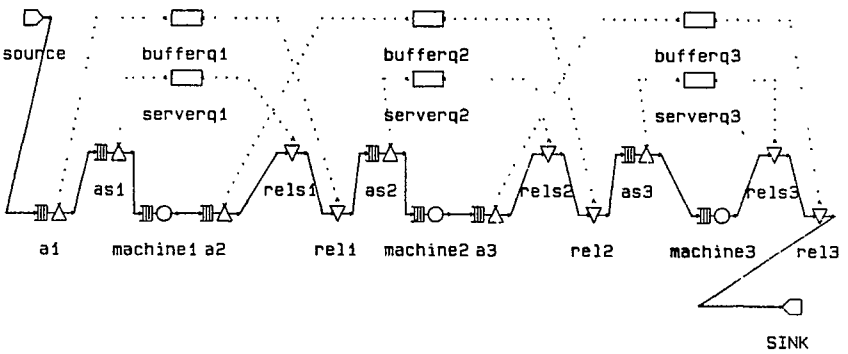

Figure 3: Continuous Flow Manufacturing System

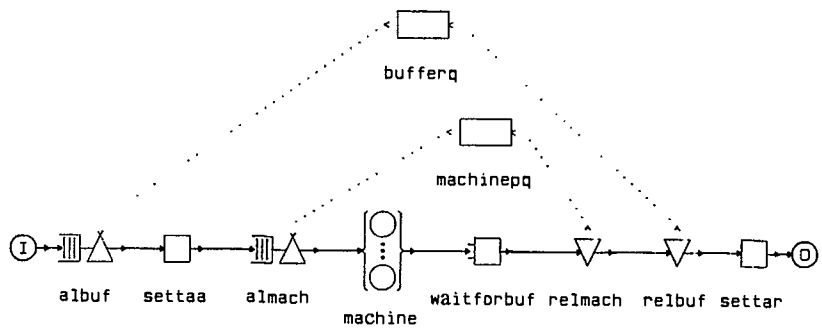

Figure 4: Diagram of Submodel for Finite Buffers

this and truly model a continuous flow manufacturing system. This second passive queue will have one token representing the job in service. That job will only release its token (and thereby allow the next job to be serviced) when it can get space at the next machine's buffer. Figure 3 shows the resulting network diagram for the continuous flow manufacturing system.

4. A SUBMODEL FOR FINITE BUFFERS

Instead of having to create the above overlapping passive queues and service centers to represent each machine in a pull system, we would like to create a submodel that can be easily incorporated in a user's model and invoked whenever a machine with a finite buffer is required. A difficulty in using submodels with the previous approach is that the boundary of such a submodel necessarily overlaps the boundaries of other nodes in the network, since the job must first be able to access the next machine before releasing its current machine. In this section, we construct a general submodel to represent machines with finite capacities and blocking which avoids the overlapping of nodes of passive queues and can be easily incorporated into a RESQ model.

To construct a submodel, we select the screen management item called layer. It allows us to move up and down in a tree of hierarchical submodels. Selecting layer gives us a new modeling screen to construct a submodel below the main model. The submodel to represent a machine with a finite buffer has to perform three types of processing: (1) allocate the buffer for the current machine, (2) allocate the current machine, perform the processing on the current machine and wait until the buffer for the next machine is available, and (3) release the current machine and its buffer when able to move to the next machine.

To accomplish this, we introduce three additional RESQ modeling elements: the set node, the wait node and the infinite server. The set node is used to assign values to variables when a job reaches that node. We use two set nodes in this submodel to set a global variable equal to the buffer size available for a given machine. This is performed when a machine's buffer is allocated and when it is released. This information will be used at a wait node to test for the availability of the buffer at the next machine. The wait node enables us to avoid overlapping the boundaries of different invocations of the submodel. The infinite server is a service center with a sufficient number of servers to ensure that no job will have to wait for service. We use this type of node to represent the machines in this submodel, since a job can only enter that node when a machine is free. In the icon palette shown in Figure 1, the set node is the icon in the third row, first column, the wait node is the icon in the first row, fifth column, and the infinite server is represented by the icon in the second row, second column.

Figure 4 is a diagram of the submodel. This submodel is invoked for each machine in the manufacturing line example with appropriate values for the service time and buffer size parameters. When a job reaches the submodel's input, its route consists of allocating the buffer for the current machine, setting the global variable equal to the number of buffers available, allocating a machine, performing the processing on the machine, possibly waiting for a buffer to be available at the next machine, releasing the machine and the buffer for the current machine and setting the global variable equal to the number of buffers available. The job then leaves the submodel to advance to the next machine.

Figure 5 is a diagram of the main model. This same submodel can be used when routing to parallel machines, as well as to serial machines as illustrated above.

5. HIGH-LEVEL MODELING CONSTRUCTS

The submodel discussed in the previous section is just one example of a higher-level modeling construct that would be useful in modeling manufacturing systems. To allow the engineer/planner to more easily build manufacturing models, we propose the development of higher-level building blocks

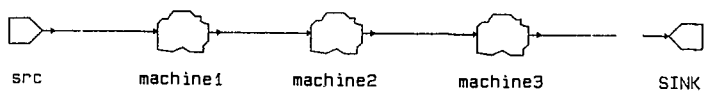

Figure 5: Diagram of Higher-Level Modeling

which can be linked together to form a complex manufacturing model. At the same time, the engineer/planner has access to all the low-level constructs of RESQ. The higher-level building blocks will be RESQ submodels that are represented by user-defined icons. Some of the manufacturing building blocks that we are considering to implement include:

- Material handling system components
  - Robots
  - Automated guided vehicles
  - Automated storage and retrieval systems
  - Manual-guided equipment
  - Conveyor systems
- Merging and unmerging of jobs
- Batch arrivals and service
- Equipment down time
- Operator availability

The following describes the high-level building component design for RESQME. The modeler creates a submodel as in Section 4. He then requests the icon-drawing package and is presented with a grid in which he draws an icon to represent that submodel. He links the icon to the submodel and stores the resulting construct. The user-created icon is then added to the icon palette and can be selected along with other user-created icons as well as lower-level icons to construct a manufacturing system.

When a user-created icon is selected and placed on the modeling surface, the corresponding submodel template appears on the character screen, allowing the user to enter any submodel parameter values needed. The user-created icon has an input and output port to connect to other icons. Figure 5 shows the manufacturing line for the pull system with the user-created icon for the submodel described in Section 4.

By selecting modify and picking that user-created icon in the icon palette, the user can view/modify the details of the submodel which consists of its diagram of nodes and routing and its underlying attributes.

Because the high-level building blocks are RESQ submodels and because the RESQ elements themselves are also available, this design has unique flexibility and power.

6. SUMMARY AND FUTURE DIRECTIONS

We have described the construction of a model of a pull system using the graphical interface and hierarchical modeling capabilities of RESQME. We used this as a basis to discuss one example of a higher-level modeling construct that can be used to represent some complex manufacturing processes. This design makes the visual programming capabilities of RESQME extensible by allowing users to create and combine their own constructs. We have also indicated some other high-level building blocks which we intend to implement for use by manufacturing engineers and planners.

In addition to the above work, we intend to enhance RESQME with an animation capability, a work unit management component to allow the user to access and maintain all the files related to a family of models, a parametric analysis facility to efficiently investigate model solutions over a large parameter space, and dynamic tutorials to teach users about the system, simulation methodology and the RESQ constructs.

ACKNOWLEDGEMENTS

We would like to express our continuing thanks to Peter Welch for his support and encouragement of this work, to Richard Gilbert and to Mark Giampapa for their technical advice, and to Al Blum, Paul Loewner and David Stein for their many suggestions which are helping to improve the package. We would also like to express our appreciation to We-Min Chow, Darcy Hulseman and Howard Jachter for their contributions to functions of the high-level modeling system. In conversations with them they have contributed many ideas for future directions of RESQ. We would also like to acknowledge the continuing interest of Charles Sauer in RESQ. His contributions to RESQ continue to be of tremendous benefit. We are grateful to the many other 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.

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

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

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 Conference*, 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.

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., 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. (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.

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 is 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 704
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 704
Yorktown Heights, NY 10598
(914) 789-7561

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