Proceedings of the 1985 Winter Simulation Conference
D. Gantz, G. Blais, S. Solomon (eds.)

AN INTRODUCTION TO THE RESEARCH QUEUEING PACKAGE

Edward A. MacNair
IBM Thomas J. Watson Research Center
Yorktown Heights, New York 10598

Abstract: The Research Queueing Package (RESQ) is a general purpose modeling tool for constructing and solving extended queueing network models of computer systems, communication networks, automated manufacturing systems and other types of contention systems. This paper is an introduction to RESQ which describes the basic model elements, submodels, statistical output analysis and graphical results.

1. INTRODUCTION

The Research Queueing Package (RESQ) [5,6,9-16] is a collection of programs for constructing and solving extended queueing networks models. Queueing network models [2,4,8] have been used to solve models of computer systems, communication networks and manufacturing systems for many years. Extensions to queueing network models permit more realistic representations of complex contention systems. RESQ contains a set of extensions in the form of several high level modeling elements. RESQ models can be solved analytically if they satisfy product form restrictions. If these restrictions are violated, simulation can be used to produce the performance measures.

The solution techniques employ state-of-the-art methods for solving the models. The analytic technique is the Mean Value Analysis algorithm [7]. The discrete event simulation program employs several methods to produce confidence intervals. Sequential stopping rules and run length determination [3] are also used.

Computer systems contain many complex resources and algorithms. RESQ can be used to model many of these complexities. Modeling elements are available for representing memory constraints and allocation, parallel operations, channel, control unit, head of string and I/O device contention, various scheduling algorithms including priorities, buffered devices, multiple workloads, synchronization of independent tasks, multiple processors, buses and much more.

Communication networks are just as complicated. RESQ can be used to study transmission lines, long haul networks, local area networks, polling, pacing, flow control, finite buffer space, store and forward networks, transmission control units, remote terminals, acknowledgements, time outs, packetizing of messages, adaptive routing, CSMA/CD and token protocols and many more features of these systems.

Automated manufacturing systems are also very complicated systems. Modeling elements available with RESQ can depict tools, machines, robots, bulk arrival of jobs, machine failures, rework, work in progress, buffers, merging and splitting of tasks, storage areas, orientation procedures, batching of jobs, transfer units, conveyor units and other items.

Section 2 discusses the basic model elements of RESQ. Submodels, which simplify model construction, are presented in Section 3. The statistical output analysis performed by RESQ is described in Section 4. Section 5 is a brief explanation of the types of graphical results which are available with the package. Section 6 illustrates a simple RESQ example. The last section is a short conclusion.

2. BASIC MODEL ELEMENTS

Contention systems are collections of interconnected resources through which basic entities (jobs) flow and demand service. RESQ provides extensive facilities for representing both the resource network and the behavior of jobs within that network. Models can be solved by simulation, or where the model structure, parameters, and assumptions permit, by an analytical method.

In RESQ, system resources are represented by "queues." Queues are grouped into two general categories, "active" and "passive." Jobs in active

queues cannot interact with other model elements while remaining in the queue. Those which visit passive queues can simultaneously occupy other system resources and perform other activities. Active queues are further subdivided into types based upon their service disciplines: first come first served (FCFS), last come first served (LCFS), priority (PRTY), preemptive priority (PRTYPR), infinite server (IS), processor shared (PS), and a generic queue type called ACTIVE. This last type permits representation of active queues (such as multiple server, shortest or longest remaining time first, SRTF or LRTF) for which there is no predefined RESQ queue entity. Associated with active queues are one or more "classes" (waiting lines) which may be used to distinguish jobs with different service time distributions and, where appropriate, priorities.

Passive queues are used to model concurrency of events. Each passive queue has an associated "token pool." When jobs arrive at a passive queue, the RESQ simulation program attempts to allocate a user-specified number of tokens from the associated token pool. If the pool does not contain the required tokens, the job is delayed until they become available. Passive queues are often used for modeling contention for shared resources such as computer memory, input/output channels, controllers, etc. Response time measurements and the representation of communication protocols are two additional typical uses for passive queues.

The structure of a model is defined by linking resources together by means of "chains." Chains designate the permissible paths over which jobs can be routed. Chains are either "closed" or "open," and also either "external" or "internal." Closed chains have a fixed "population" of jobs which circulate in the chain for the entire model execution time. In general, open chanins contain a variable number of jobs that are generated at input "sources" to the chain and terminated at "sinks." External and internal chains are used with "submodels," which are parameterized templates of user defined subsystems. External chains are connected to chains within the exterior model (or submodel) that invokes the submodel. Internal chains on the other hand are contained entirely within a submodel. Both external and internal chains can be either open or closed; however, an external chain is open (closed) if the chain to which it is connected is open (closed).

Several RESQ entities are provided to add flexibility in modeling alternative timings, configurations, job routing, etc. "Numeric" and "distribution parameters" can be modified before each solution of a model to test the effect of variations in arrival and service rates and distributions, number of users, memory size, etc. "Job variables" are associated with each job to define its behavioral characteristics, such as its "type", memory and service requirements, and routing decisions. "Chain variables" can be identified with each chain and are accessible only to jobs within that chain. "Global variables" may be accessed from anywhere within their defined (model or submodel) scopes to allow for communication between jobs and other RESQ entities. Job, chain, and global variables are assigned values at "set" nodes by "expressions" with rules closely resembling those found in most programming languages. In RESQ simulation models, expressions containing job, chain, and global variables, can be used to model status dependent assignments and decisions.

Some other RESQ entities are "split", "fission", and "fusion" nodes which permit the splitting of jobs into copies, another method for modeling simultaneity as well as signaling between, and synchronization of, processes. The split and fission nodes create additional copies of a job passing through the node; a fusion node joins those jobs that have previously separated at a fission node.

A variety of measures are available for evaluating system performance. Some of these include resource utilization and throughput, mean queue length, queue length distribution, mean queueing time, and queueing time distribution. Similar output is also available for the usage of passive queue tokens.

3. SUBMODELS

A submodel is a portion of a model containing parameters which can be assigned values. A submodel may contain any subset of resources present in the model, and we may make one or more copies of the submodel. One restriction is that a queue definition must be entirely contained within of a submodel. Therefore, a queue may not be partially defined in one submodel and the remainder defined in another submodel or the outer model. Submodels can be used to clarify the structure of a model, to avoid duplication of effort within a model, to permit sharing of parts of models, to introduce variability in the model structure and, with decomposition, to solve the submodel separately and replace the submodel with a flow equivalent server.

The structure of the model can be clarified by constructing submodels for the major subsystems to be represented. The submodels can be used to represent high level abstractions of the subsystems which can be easily connected to form the overall system. If we have a model of a system which has a CPU and an I/O subsystem, we could construct a submodel representing the CPU and another submodel representing the I/O subsystem. The I/O could also be decomposed into submodels nested within it representing each I/O device.

If a model contains subsystems which are similar, we could construct a submodel representing one copy of the subsystem with parameters which will capture the differences. Then the submodel can be duplicated for each subsystem with different values supplied for each copy of the submodel. In a communication network with several similar host computers, a submodel representing one host could be constructed and easily duplicated for each host needed in the model.

Many models contain subsystems which are similar to those used in other models. Submodels faciliate the use of portions of models. If a submodel of a CPU with round-robin scheduling has been constructed, this submodel can be used in many different models.

Hierarchical decomposition is a widely used technique for simplifying the solution of certain types of models. The model is decomposed into one or more submodels which are solved separately without the remainder of the model. Results from the submodel solution are used to characterize a flow equivalent

server which is used in place of the submodel in an aggregate model. The flow equivalent server is usually a queue dependent server with the service rates a function of the queue size. This decomposition and substitution can be accomplished by using any solution technique for solving the submodels and the aggregate model.

Very frequently models have a requirement for having a variable number of resources. The number of resources can be specified as a model parameter, and a submodel can be built to represent one of the resources. RESQ permits an arbitrary number of copies of the submodel to be created based on the value of a model parameter. A simple example of this is a model of an I/O subsystem which we want to evaluate for a variable number of I/O devices. This can be easily represented by using a submodel of one device and making a variable number of copies of it using an invocation array. Node and chain arrays are also useful for introducing variability in the model structure.

4. STATISTICAL OUTPUT ANALYSIS

Since simulation is equivalent to a statistical sampling experiment, RESQ makes available three methods for generating confidence intervals for performance measurements. Under the "independent replications" method, the simulation is executed several (user specified) times with different sequences of random numbers. An "initialization" period may be specified for each replication; data collected during this period will be discarded when calculating output statistics. The effect of transient results may thus be minimized, permitting model equilibrium to be better approximated. A confidence interval with a (user) specified confidence level is then calculated after all replications have been completed. The "spectral" confidence interval method [1] uses the correlation properties of sequential values of the measured parameter instead of relying solely upon the assumption of independence. It too provides options for specifying a confidence level, run limits, and initialization period, as well as a sequential stopping rule whereby the user designates a confidence interval width criterion for terminating the simulation. The "regenerative" method [4,8] has some very special requirements for its use. A state in the model must be identified at which the system "regenerates." That is, the future behavior of the system is independent of all states prior to entrance into the regeneration state. The regenerative method provides all the options of the spectral method except for intialization period specification.

5. PARAMETRIC SOLUTIONS AND GRAPHICAL RESULTS

RESQ can be used to perform multiple solutions of a model, possibly over a large parameter space. The solutions can be accomplished using an analytic method or by simulation. The data extracted from these solutions will normally be passed to the next phase which plots the data. Part of the interactive dialogue associated with performing the parametric solutions is related to the grouping of the data for subsequent plots. The data can be grouped into multiple plots, and each plot may contain multiple curves on it.

The points on the x-axis of each curve represent information from different model solutions. Corresponding points from different curves can contain results from the same model solutions or from different model solutions. Model solutions with different parameter values are specified with the parametric solution command by using a numeric parameter as a variable for the X-axis values of the plots. The values of the numeric parameters associated with the axes can be specified as a list of values or in a type of loop notation. The loop notation provides a starting value, an ending value and an optional increment. Other model parameters which are not associated with either axes are also given values. Their values will remain fixed for all of the solutions.

Each point to be plotted can be specified by an arithmetic expression involving RESQ performance measures and constants. An expression can simply refer directly to a standard RESQ performance measure, like the utilization, throughput, queue length or queueing time. It can also combine performance measures, numeric parameters and constants with arithmetic operations to calculate results which are not normally available.

The command for performing the parametric solutions is interactive and prompts the user for the required information for solving a specified model. The usual RESQ results from each solution may be examined. The command produces a file containing the interactive dialogue seen at the terminal and a file of data to be plotted. The plotting file is provided as a link to the next phase, but this file can also be used by other plotting packages. In addition to answering the prompts interactively, the user can supply a file containing some or all of the responses. This can reduce the necessity of answering the questions interactively and provides for ease of repeatability and future changes.

The graphical display of the results is produced in the second phase. There are several reasons for separating the two phases. The main reasons are for ease of revising the plots without having to solve the model again and to provide a means of using other plotting routines. The RESQ plotting command provides an interactive dialogue mode for specifying the format of each plot.

The user can specify the number of plots to display on the same screen. For each plot, minimum and maximum coordinate values can be specified. If any of these are not given, they are determined from the data. Plot labels can be given. These include a plot heading, a legend for each curve, a color for each curve, coordinate labels and the use of linear or logarithmic plotting scales. Since the plotting is independent of the solution phase, the plots can be reviewed and easily modified at little computational expense.

Just as with the evaluation phase, the plotting phase produces a file as input containing the interactive dialogue of the plots produced and a file containing the plots themselves. The plotting command can also use a file as input which contains some or all of the responses for the prompts. There are default values for most of these prompts, so a null response can be given when the default is desired.

6. EXAMPLE

In this section we present a simple example to illustrate some of the features of RESQ. It is a model of an interactive computer system. The SETUP command, which is used for constructing the model, can be used in an interactive fashion. The characters to the left of a colon are a prompt, and the information on the right of a colon is the user's response.

The program first prompts for the solution method and then the queue definitions. Active queues are used for the CPU, I/O devices and the terminals. A passive queue is used to represent memory contention.

```
setup csmtm
   METHOD: simulation
   QUEUE: cpuq
     TYPE: active
     DSPL: ps    /* processor sharing */
     CLASS LIST: cpu
        WORK DEMANDS: .004
     SERVER-
        RATES: 0.2
   QUEUE: disk1q
     TYPE: fcfs
     CLASS LIST; disk1
        SERVICE TIMES: .044
   QUEUE: disk2q
     TYPE: fcfs
     CLASS LIST: disk2
        SERVICE TIMES: .008
   QUEUE: terminalsq
     TYPE: is
     CLASS LIST: terminals
        SERVICE TIMES: 5
   QUEUE: memoryq
     TYPE: passive
     TOKENS: 4    /* memory partitions */
     DSPL: fcfs
     ALLOCATE NODE LIST: allocmem
        NUMBER OF TOKENS TO ALLOCATE: 1
     RELEASE NODE LIST: relsemem
```

Next comes the chain definition. We are using a closed chain with the number of customers equal to the number of terminals. The routing information specifies how the jobs move. The two characters -> are used to represent an arrow for transtions from one node to another. Notice that arithmetic expressions can be used where a number is expected.

```
   CHAIN: chain1
     TYPE: closed
     POPULATION: 15
     : terminals->allocmem->cpu
     : cpu->disk1 disk2;.2 .8
     : disk1->cpu relsemem;19/20 1/20
     : disk2->cpu relsemem;19/20 1/20
     : relsemem->terminals
```

The end of a simulation model contains some information about performance measure distributions, confidence interval method and run length.

```
   QUEUES FOR QUEUEING TIME DIST: memoryq
     VALUES: 3.5
   CONFIDENCE INTERVAL METHOD: regenerative
   REGENERATION STATE DEFINITION-
   CHAIN: chain1
     NODE LIST: terminals
        REGEN POP: 15
        INIT POP: 15
   CONFIDENCE LEVEL: 90
   SEQUENTIAL STOPPING RULE: yes
     QUEUES TO BE CHECKED: memoryq
        MEASURES: qt
        ALLOWED WIDTHS: 10  /* percent */
     SAMPLING PERIOD GUIDELINES-
        SIMULATED TIME: 7200
        QUEUES FOR DEPARTURE COUNTS: memoryq
          DEPARTURES: 1000
   LIMIT-CP SECONDS: 12
   TRACE: no
 END
```

The following results were obtained for this model. After the simulation stops, some summary statistics are produced. This is a very short run, and some of the confidence intervals are not very accurate.

```
eval csmtm
RESQ2 VERSION DATE: MARCH 11, 1985 -
   TIME: 19:30:50  DATE: 03/19/85
SAMPLING PERIOD END:
   MEMORYQ DEPARTURE GUIDELINE
RUN END: CPU LIMIT
NO ERRORS DETECTED DURING SIMULATION.
   303 DISCARDED EVENTS

          SIMULATED TIME:     480.37329
                CPU TIME:         12.03
        NUMBER OF EVENTS:         41553
        NUMBER OF CYCLES:            68
```

Following the summary statistics, any or all of the performance measures may be examined. Here we display most of the interesting results.

```
WHAT: allbo

ELEMENT      UTILIZATION
MEMORYQ      0.78078(0.73426,0.82730) 9.3%
CPUQ         0.84243(0.81367,0.87118) 5.8%
DISK1Q       0.37477(0.35695,0.39258) 3.6%
DISK2Q       0.27064(0.26084,0.28044) 2.0%
TERMINALSQ   0.00000(0.00000,0.00000)

ELEMENT      THROUGHPUT
MEMORYQ      2.18372(2.10055,2.26688) 7.6%
CPUQ         42.1588(40.7757,43.5419) 6.6%
DISK1Q       8.44135(8.05874,8.82396) 9.1%
DISK2Q       33.7175(32.5968,34.8381) 6.6%
TERMINALSQ   2.18372(2.10055,2.26688) 7.6%
RELSEMEM     2.18372

ELEMENT      MEAN QUEUE LENGTH
MEMORYQ      4.47174(4.00841,4.93508) 20.7%
CPUQ         2.20322(2.05684,2.34960) 13.3%
DISK1Q       0.56167(0.52065,0.60268) 14.6%
DISK2Q       0.35824(0.34142,0.37507) 9.4%
TERMINALSQ   10.5282(10.0649,10.9915) 8.8%

ELEMENT      STAND. DEV. OF QUEUE LENGTH
MEMORYQ      2.65582
CPUQ         1.38729
DISK1Q       0.86496
DISK2Q       0.67021
TERMINALSQ   2.65582
```

```
ELEMENT        MEAN QUEUEING TIME
MEMORYQ        2.04777(1.82202,2.27351) 22.0%
CPUQ           0.05226(0.04998,0.05454) 8.7%
DISK1Q         0.06654(0.06345,0.06962) 9.3%
DISK2Q         0.01062(0.01042,0.01083) 3.9%
TERMINALSQ     4.82125(4.54008,5.10242) 11.7%

ELEMENT        STAND. DEV. OF QUEUEING TIME
MEMORYQ        1.76652
CPUQ           0.05813
DISK1Q         0.06730
DISK2Q         0.01063
TERMINALSQ     4.69089

ELEMENT        MEAN TOKENS IN USE
MEMORYQ        3.12313(2.93705,3.30921) 11.9%

ELEMENT        MEAN TOTAL TOKENS IN POOL
MEMORYQ        4.00000

ELEMENT        QUEUEING TIME DISTRIBUTION
MEMORYQ      3:0.76454(0.72093,0.80815) 8.7%
             5:0.93804(0.92105,0.95502) 3.4%

ELEMENT        MAXIMUM QUEUE LENGTH
MEMORYQ        12
CPUQ           4
DISK1Q         4
DISK2Q         4
TERMINALSQ     15

ELEMENT        MAXIMUM QUEUEING TIME
MEMORYQ        11.11583
CPUQ           0.56125
DISK1Q         0.50105
DISK2Q         0.09635
TERMINALSQ     38.15923

WHAT:
CONTINUE RUN: no
```

7. CONCLUSION

RESQ provides efficient and accurate model solutions with very useful simulation output analysis. It facilitates the accurate representation of systems with extended queueing network model elements. The user interfaces increase productivity through interactive and file modes of model definitions, submodels and interactive simulation.

ACKNOWLEDGEMENT

We are particularly appreciative of the contributions of Charles Sauer to this work. Most of the concepts and some of the examples in this report are his, and the reader can assume in most cases they were originated by Sauer, with some assistance from us. We are grateful to the many colleagues and RESQ users who have helped us in this work and contributed to RESQ in general.

REFERENCES

1. P. Heidelberger and P.D. Welch, "A Spectral Method for Confidence Interval Generation and Run Length Control in Simulations," CACM 24,4, April 1981, pp. 233-245.

2. H. Kobayashi, Modeling and Analysis: An Introduction to System Performance Evaluation Methodology, Addison-Wesley, 1978.

3. S.S. Lavenberg and C. H. Sauer, "Sequential Stopping Rules for the Regenerative Method of Simulation," IBM J. of Research and Development 21,6, November 1977, pp. 545-556.

4. S.S. Lavenberg, Editor, Computer Performance Modeling Handbook, Academic Press, New York, 1983.

5. E.A. MacNair and C. H. Sauer, "The Research Queueing Package: A Primer." Proceedings of SHARE60, February 1983, pp. 29-37.

6. E.A. MacNair and C. H. Sauer, Elements of Practical Performance Modeling, to be published by Prentice-Hall, Englewood Cliffs, N.J., 1985.

7. M. Reiser and S.S. Lavenberg, "Mean Value Analysis of Closed Multichain Queueing Networks," JACM 27, 2, April 1980, pp. 313-322.

8. C.H. Sauer and K.M. Chandy, Computer Systems Performance Modeling, Prentice-Hall, Englewood Cliffs, N.J., 1981.

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

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

11. C.H. Sauer, E.A. MacNair and J.F. Kurose, "The Research Queueing Package Version2: Introduction and Examples;" IBM Research Report RA-138, Yorktown Heights, New York, April 1982.

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

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

14. C.H. Sauer, E.A. MacNair and J.F. Kurose, "The Research Queueing Package: Past, Present and Future," Proceedings of the National Computer Conference, June 1982, pp. 273-280.

15. C.H. Sauer, E.A. MacNair and J.F. Kurose, "Queueing Network Simulations of Computer Communication," IEEE Journal on Selected Areas in Communications, Vol. SAC-2, 1, January 1984, pp. 203-219.

16. C.H. Sauer, E.A. MacNair and S. Salza, "A Language for Extended Queueing Network Models," IBM J. of Research and Development 24, 6, November 1980, pp. 747-755.

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 in the Modeling and Analysis Software Systems project developing modeling programs to solve extended queueing networks. In addition, he is an adjunct staff member at the IBM Systems Research Institute, where he teaches a course 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, to be published by Prentice-Hall, 1985. He holds an M.S. degree in Operations Research from New York University and is a member of ACM and ORSA.

Dept. 541, Loc. 81-S03
Thomas J. Watson Research Center
P. O. Box 218
Yorktown Heights, NY 10598
(914) 945-1447