Proceedings of the 1990 Winter Simulation Conference
Osman Balci, Randall P. Sadowski, Richard E. Nance (eds.)

A GENERALIZED RELIABILITY BLOCK DIAGRAM (RBD) SIMULATION

Kerry D. Figiel

International Paper Company
P.O. Box 312
Bastrop, Louisiana 71221-0312

Dileep R. Sule

Mechanical/Industrial Engineering Department
Louisiana Tech University
P.O. Box 10348
Ruston, Louisiana 71272-0046

ABSTRACT

One of the major problems encountered in designing a general purpose simulation model is the establishment of a structure which links the various processes together in a way that does not require customized programming. This paper demonstrates a computer simulation yielding reliability and maintainability information for any system based upon the failure and repair distributions of the individual components. The information needed to link the various components together is taken directly from the Reliability Block Diagram (RBD), thereby negating the need for writing specialized programs. Final results display 90% confidence intervals for 4 key endogenous variables, System Availability, First Failure Time, Mean Time Between Failure (MTBF) and Mean Time To Repair (MTTR).

Techniques for reducing computer memory requirements are discussed. The paper demonstrates how connections between process blocks can be generalized and linked together during program execution. This method results in more useful programs that solve a broader range of problems.

1. INTRODUCTION

Kapur and Lamberson (1977) used Reliability Block Diagrams (RBDs) to describe the possible combinations of operational components that will result in total system reliability. The components of the system are the Reliability Blocks that can be characterized by probability distributions for failure and repair. The connections between Reliability Blocks identify the success paths for a system, or the various combinations of working components that can result in an operational system. A component is on an operational path if at least one path can be traced from the beginning of the RBD to that component through non-failed components. The system is therefore operational if there exists at least one operational path between the beginning and end of the RBD. Within the RBD, components may be organized in parallel, series or other more complex relationships including various levels of redundancy.

Certain techniques already exist for determining the probability of system reliability under certain circumstances. If the reliability of each component is specified over some established time-frame, a static model is formed. System reliability (the probability that the system will remain operational) can then be calculated using a number of techniques including series and parallel analysis, the event space method, path tracing, decomposition, cut sets and tie sets. Repair activities are not modeled in these static models and no variance information is provided.

Dynamic models incorporate time-dependent functions into the reliability model by deriving an overall system hazard function from the hazard functions of the individual components. Whereas the calculation of the overall hazard function is relatively straightforward for exponential failure rates, other distributions can greatly complicate the analysis. Repair activities still are not included in the dynamic model and other system effectiveness measures such as availability and maintainability are not provided. To obtain these additional measures of system effectiveness, stochastic means are normally applied.

To fully understand a complex system, not only must the failures be modeled, but also the repair activities. The above mentioned deterministic techniques do not take into account activities that can repair a failed component before it affects system reliability. This complexity is difficult to model other than in simulation. Additionally, the variability of the endogenous variables is a key item of interest. For example, one may be interested not only that System Availability averages 95%, but whether it is steady and consistent at 94% - 96% or ranges widely between 83% - 99.9%. As failure and repair activities can be of different distribution types (uniform, normal, exponential, log normal, etc.), deterministic means are difficult to apply. Again simulation gives one the ability to deal with these complexities and evaluate a number of simulation runs to determine variability and confidence intervals.

The immediate problem in constructing a general purpose simulation model is how one can efficiently describe all possible connection of blocks in limited computer memory. RBD's by their nature require a sequential arrangement of blocks without loopbacks. In other words the success path to a block cannot depend on the operational status of the block itself. Consequently, within a system of N blocks, any of the N may be chosen as the starting point. The first block may connect to any or all of the remaining N-1 blocks but not to itself. The second block chosen in sequence may connect to any of the remaining N-2, but not to the first or itself. If each block were free to connect to any other block, N2 paths would be possible (i.e. N components connecting to N components). As loopbacks are not allowed, the maximum number of possible connections for each block can be viewed as the decreasing series, N-1, N-2, N-3 ,..., 3, 2, 1 as freedom is restricted with each subsequent choice. The maximum number of possible connections in any RBD is therefore the sum of this series or $N(N-1)/2 = N^2/2 - N/2$. As the number of possible connections is a function of the square of the number of blocks, computer memory requirements become exorbitant as for large block size models.

2. THE MODEL

The solution to this problem is the development of a matrix structure which keeps track of the connectivity of the blocks (the output connections) and the operational status of paths and individual blocks. Five variables for each block are maintained to implement this solution. They are:

1) Input threshold -Required operational paths to block?
2) Block input status -Actual operational paths to block?
3) Block operational status -Is the block itself operational?
4) Block output status -Operational path to and through block?
5) Output connections -Increments input status of next blocks.

The status of the system is determined by processing all blocks sequentially when any block changes state (next event simulation). To process the above block model, one first checks whether the number of operational inputs to a block (the input status) satisfies the established threshold. If so, then a valid operational path exists to the block. Normally, one valid input is needed for a valid operational path to exist. However, certain circumstances such as redundancy may require some higher value. If four paths connect to a block and two are required for the system to function, this block's input threshold would be set to 2.

After one has established that the block is indeed on an operational path, the block's operational status governs whether that operational path will be continued and linked to the output blocks. When a block is operational and has satisfied its input

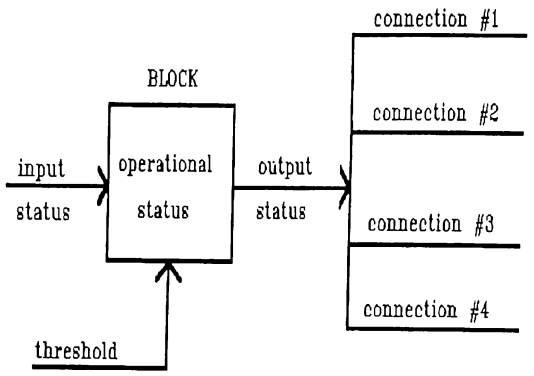

Figure 1. Block Model

threshold, the block's output status word is set. Linking is carried out by incrementing the input status words of all connected blocks. When it is then the connected blocks turn to be processed, input connections and operational status will again be checked and new linkages performed.

As blocks are processed in sequential order, it is required that no block have an output connection to a lower numbered block. A connection will be defined to be uni-directional from the output of one block to the input of the next. Block 1 therefore must always be the starting point of this RBD model and has an operational input status by default. The output status of the last block represents the System's operational status, and is the principle endogenous variable of interest.

To design a program that will fit in memory, some heuristic limit to the number of connections per block must be made. This allows the dimensioning of the connection in the matrix structure to some finite value. As already noted, the array size needed to handle the model becomes unmanageable when no such limitation is made. It is therefore required that the number of connections per block be limited to some reasonable value to economize on memory requirements.

Dummy blocks can be added to the model to provide full connectivity. As dummy blocks will never be allowed to fail within the time frame of the simulation, they can never contribute to system downtime, and are transparent to the results obtained. Hence, the number of output connections per block can be effectively expanded without wasting vast array space on improbable combinations or losing generality. In this particular model a four connection per block limit was selected as a prudent compromise.

Figure 2 shows how a reliability block structure allowing only four output connections per block can be effectively expanded. In this example, Block B001s output status is connected to eight other blocks (B004 through B011) via the use of two dummy block connections (B002 and B003).

In comparing the memory requirements for a 100 block model, one finds that without this technique, 100(99)/2 = 4950 memory locations would be required to allow for all possible connections. With the 4 connections per block model, only 4*100= 400 memory locations are needed if no Dummy Blocks are used. Dummy Blocks do consume blocks that are otherwise available for use and in that sense are an added memory cost. In the computer program used in this paper, 10 integer and 10 reals variables are used for record keeping on each block. Assuming that each real value consumes two memory locations (double word reals are very common in the computer industry), one finds that 30 memory locations are required for each Dummy Block. Hence the tradeoff in size can be easily evaluated. The technique saves 4950-400 = 4450 memory locations, which is equivalent to 4450/30 = 148 dummy blocks. Consequently, it is more efficient to use the 4 connection per block model on a 100 component system as long as the number of Dummy Blocks does not exceed 148. In practice, 4 output connections per block is adequate for most RBDs. It should be noted, however, that the choice of 4 output connections per block is heuristic in nature and certain situations may be better suited to a larger or smaller number.

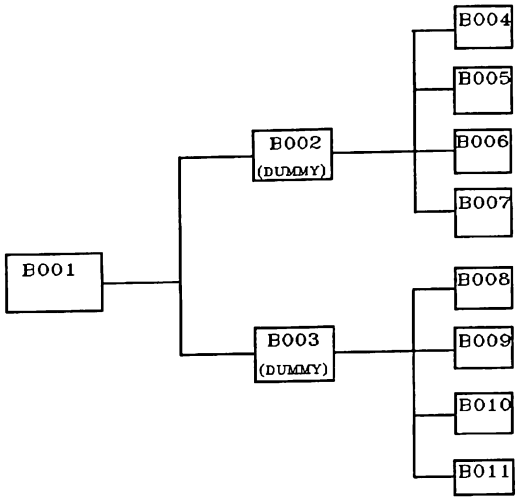

Figure 2. Use of Dummy Blocks to Expand Output Connections

3. SIMULATION PROGRAM

A simulation program was written to implement this general RBD model. It was written in Fortran 77 to run on an Harris H1200 super mini-computer, but may be easily modified to run on any of a wide variety of computers having Fortran compilers. Input and output to the program are via text files specified by the user. File input is selected over interactive input to reduce errors by allowing the user to study and edit the file. It is very discouraging for a user to complete a long simulation run only to discover that the input was entered incorrectly. The user may optionally elect to enable a trace print out that records key system variables after each event. The trace print out is, however, I/O intensive and slows the simulation run. Key element of the program are:

3.1 Inputs

The input file to the program is divided into three sections. Section 1 provides general information needed by the simulation model, i.e. the time limit (in simulation minutes), a logical entry specifying whether the trace features of the program are to be enables, a second logical entry to determine if interim results are to be printed after each simulation run, and the starting random number seed.

```
EXAMPLE OF SECTION ONE:
10000.0 FALSE TRUE  28923
END
```

The interpretation of this line is to run the simulation for 10,000 minutes of simulation time, disable trace statements, enable interim results, and use a starting random number seed of 28923.

Section 2 is taken from the RBD and defines all blocks used, up to four output connections and the blocks' input thresholds. These lines are prefaced by the specifier B for a Block Diagram Definition.

EXAMPLE OF SECTION TWO LINE:
B010 12 14 17 0 2

The interpretation of this line is that block B010 is to connect to three other blocks (B012, B014 and B017) and is to have an input threshold of 2. A zero or blank for an output connection implies an unused connection. A zero or blank threshold defaults to a value of 1. Hence, B010 12 means that block B010 connects only to block B012 and has an input threshold of 1.

Section 3 defines the repair and failure distributions associated with each block that will be used in the simulation. The R specifier is used to denote a repair distribution while the F denotes a failure distribution. The following table is used to identify the allowable distributions and their associated parameters.

TYPE	PROCESS	PARAMETER 1	PARAMETER 2
1	Constant	Value	
2	Exponential	Mean	
3	Normal	Mean	Standard Deviation
4	Uniform	Mean	Standard Deviation
5	Log Normal	Mean	Standard Deviation

Table 1. Repair and Failure Distribution Parameters

EXAMPLE OF SECTION 3 INPUT:
F010 2 500.
R010 3 100. 25.

The above lines define the repair and failure distributions for block B010. Hence, B010 fails according to an exponential distribution with a mean of 500 minutes. Repair is normally distributed with a mean of 100 and standard deviation of 25 minutes. Should the failure distribution be omitted, it is assumed that the block will never fail during the simulation. Likewise, if the repair distribution is omitted, it is assumed that the block can never be repaired during the simulation. In these cases the program selects constant failure and repair processes with rates of 5 times the total simulation time. All sections are terminated with an END statement.

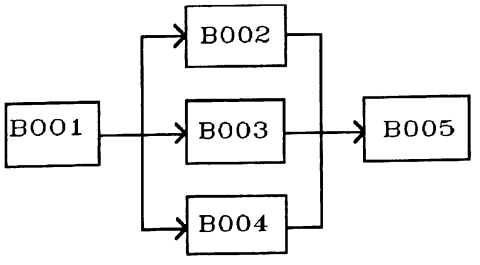

Figure 3. RBD Model of 2 of 3 Redundant System

To illustrate the input file, a simple 2 of 3 redundancy system is be modeled in Figure 3. Block B001 will be the start and B005 the termination used to measure reliability. Blocks B002, B003, and B004 each have an exponential failure rate of 500 minutes, and normal repair rate of 100 minutes with a 25 minute standard deviation. The threshold for B005 is set to 2 to make it a 2 of 3 system. The Block diagram for this system is as depicted in Figure 3.

```
1000.0   TRUE        TRUE        68923
END
B001     2     3     4     0
B002     5
B003     5
B004     5
B005     0     0     0     0     2
END
F002     2     500.
R002     3     100.        25.
F003     2     500.
R003     3     100.        25.
F004     2     500.
R004     3     100.        25.
END
```

Figure 4. Input File for 2 of 3 Redundant System

In the input file for this model, (Figure 4) Section 1 again specifies a simulation of 10,000 minutes with all trace messages and interim results enabled. A random number seed of 68923 is selected. Section 2 defines all connections. Block B001 connects to B002, B003 and B004. These blocks then in turn connect to B005. The threshold of 2 for Block B005 requires two valid paths for it to have an operational input status. Section 3 defines the distributions which apply. Blocks B001 and B005 are not modeled as they are the start and termination blocks and are therefore not permitted to fail or be repaired. B002, B003 and B004 are modeled as specified, i.e. exponential failure (type 2), normal repair (type 3).

3.2 Program Structure

The RBD program is modular in nature and uses structured programming techniques for flexibility and ease of modification. The main program itself consists of only 21 executable lines of Fortran code. Its primary function is to control the order of subroutine calls. The FOR statement is used to control the execution of 100 simulation runs to obtain the required variance information. The LOOP statement keeps each simulation running and processing "next events" until the current time exceeds the specified end time of the simulation. Consequently, the program actually performs 100 full simulations, each for the specified time limit. Once one simulation run is complete (current time exceeds the specified end time) all variables including system time are reinitialized and the next run is made with a new random number seed.

```
CALL READ INPUT FILE                  !READ ALL INPUTS
IF (ERR FLAG)                         !DID ERROR OCCUR IN READING INPUTS
  CALL PROCESS INPUT ERROR            !...YES, WRITE THE ERROR MESSAGE
  CALL EXIT                           !...AND TERMINATE PROGRAM EXECUTION
END IF                                !...NO, CONTINUE WITH SIMULATION

FOR NUM RUNS = 1,100                  !100RUNS FOR VARIANCE ANALYSIS
  CALL SYSTEM START UP                !START UP W/ALL BLOCKS OPERATIONAL
  LOOP                                !LOOP UNTIL SIMULATION TERMINATES
    CALL DETERMINE NEXT EVENT         !WHAT IS THE NEXT SYSTEM EVENT?
    CALL UPDATE BLOCK STATUS          !WHAT BLOCKS ARE OPERATIONAL?
    CALL DETERMINE SYSTEM STATUS      !IS THE SYSTEM OPERATIONAL?
    IF (CURRENT TIME .GE. END TIME)   !HAVE WE REACHED END OF SIMULATION
      EXIT LOOP                       !...YES, EXIT FROM THIS LOOP
    ENDIF                             !...NO, CONTINUE
  ENDLOOP                             !END OF SIMULATION LOOP
  IF (INTERIM RESULTS)                !IF TRACING INTERIM RESULT
    CALL WRITE INTERIM RESULTS        !WRITE ALL STATE VARIABLES
  ENDIF
ENDFOR
CALL WRITE FINAL RESULTS              !WRITE FINAL ANALYSIS OUTPUT
CALL EXIT                             !OUTPUT FILE AND END PROGRAM
END
```

Figure 5. Main Program

SUBROUTINE	FUNCTION
READ INPUT FILE	Process Users Input File. Places information in common.
PROCESS INPUT ERROR	Outputs messages if input is in error.
SYSTEM START UP	Initializes all variables.
DETERMINE NEXT EVENT	Determines which block has next change of state.
UPDATE BLOCK STATUS	Resolves blocks' operational status.
DETERMINE SYSTEM STATUS	Resolves blocks' and system's output status.
WRITE INTERIM RESULTS	Writes results of each simulation run.
WRITE FINAL RESULTS	Writes summary of 100 simulation runs.
PROCESS	Determines failure and repair time of blocks.
REAL SORT LO TO HI	Sorts real array in ascending order.

Figure 6. Subroutine Descriptions

Each subroutine used in the program performs certain well defined functions. A common is used to avoid the need for long argument lists being passed between these subroutines. Figure 6 depicts the function of some of the major subroutines used in the simulation.

3.3 Trace Enable

When enabled, the trace mechanism logs all major system events to a trace file. Initially, the input file is echoed to record the simulations understanding of the problem it is to solve. The initial status and transition times of each block are also recorded to the file. Then each ensuing subroutine that is called logs the information pertinent to its decision, including the result and reasons for that decision.

Figure 7 illustrates a sample trace made from a simulation run of the problem in Figure 3. The top of the listing is a simple restatement of the input file from Listing 1. After SYSTEM START UP FOR SIMULATION RUN 1, Block failure times are first calculated, displayed and then placed in an array of CHANGE OF STATE TIMES for each block. Note that Blocks 1 and 5 which have no specified failure distribution have been calculated to fail at 50,000 minutes (5 times the simulation time limit). This guarantees no failure of those components during the simulation

```
10000.   T           T                      68923
END
B001  2        3   4        0   0
B002  5        0   0        0   0
B003  5        0   0        0   0
B004  5        0   0        0   0
B005  0        0   0        0   2
END   0        0   0        0   0
F002  2      500.00        0.00         300.00
R002  3      100.00       25.00         300.00
F003  2      500.00       25.00           0.00
R003  3      100.00       25.00           0.00
F004  2      500.00       25.00           0.00
R004  3      100.00       25.00           0.00
END   3      100.00       25.00           0.00

SYSTEM START UP      1
NEXT  FAILURE TIME OF BLOCK     1 IS        50000.00
NEXT  FAILURE TIME OF BLOCK     2 IS         2400.82
NEXT  FAILURE TIME OF BLOCK     3 IS          955.63
NEXT  FAILURE TIME OF BLOCK     4 IS            1.73
NEXT  FAILURE TIME OF BLOCK     5 IS        50000.00
INITIAL CHANGE OF STATE TIMES
50000.000   2400.820    955.634        1.727     50000.000     10000.000

CURRENT TIME         1.73              PASS NUMBER           1
BLOCK     4 IS THE NEXT EVENT AT          1.73
EVENT IS A  FAILURE

CHANGE OF STATE TIMES
 50000.00    2400.82     955.63       1.73  50000.00
 10000.00

TIME INCREMENT   1.7269
BLOCK OPERATION TIMES
  1      1.7   2       1.7   3      1.7   4      1.7   5      1.7
SYSTEM OPERATIONAL TIME IS        1.73
STATUS OF NEXT EVENT BLOCK     4 IS NONOPERATIONAL
NEXT  REPAIR  TIME OF BLOCK    4 IS      126.63

 BLOCK STATUS ARRAYS
        INPUT    THRESHOLD       OPERATIONAL         OUTPUT
1         1          1                1                 1
2         1          1                1                 1
3         1          1                1                 1
4         1          1                0                 0
 5        2          2                1                 1
 SYSTEM STATUS IS OPERATIONAL
```

Figure 7. Trace Output from Figure 3 Simulation

run. Component 4 has the earliest failure time of 1.727 minutes. Although this failure seems rather short for a 500 minute expected failure rate, it corresponds to a random number value of 0.9965 which is not that unusual. CURRENT TIME then moves ahead to 1.727 minutes and block 4 is declared nonoperational. The repair of block 4 is determined to take 126.63 minutes. All block status arrays are traced after each transition showing each blocks input status, threshold, operational status and output status. Finally, the SYSTEM STATUS IS OPERATIONAL after this first change of state. The trace feature then continues to track each change of state in this manner. It should be noted that the trace enable feature is essentially a validation aid and should not be used during actual simulation runs. The trace increases the actual clock and CPU time required to complete the simulation by several orders of magnitude.

3.4 Interim Results

After each simulation run, the results from that individual run can be recorded in the output file. These interim results detail status information concerning each reliability block. The information maintained on each block includes the block operational time, failure time, first failure time, next change of state time and number of total failures for that block. This information is not used to build the confidence intervals for the final results section. Consequently, as these variables are then reinitialized for the next run, the data must be output or lost. Certainly, it is possible to store the interim information for later analysis; however, the memory cost is quite high. As each block variable would need to be maintained for 100 runs on 100 blocks, 10,000 locations are needed for each. Additionally, the point of the simulation is to determine system performance, not block performance. Validation that the computer simulation truly produces random variates of the correct distribution is better handled in the testing of the individual subroutines. All that the tracking of these blocks can accomplish is the revalidation of this fact. Consequently, when properly modeled the blocks will have failure and repair distributions according to the selected choices. The matter of concern is how does the whole system with unknown failure and repair distributions act. An example of this Interim Output is included as Figure 8.

INTERIM RESULTS OF RELIABILITY SIMULATION 1 PAGE 1

BLOCK	OPERATIONAL TIME	OPERATIONAL PERCENT	FAILURE TIME	FAILURE NUMBER
1	10000.00	100.00	-1.00	0
2	8547.66	85.48	2400.82	14
3	8202.87	82.03	955.63	18
4	7821.38	78.21	1.73	22
5	10000.00	100.00	-1.00	0

SYSTEM OPERATIONAL TIME	9324.840
TOTAL PERCENT SYSTEM UPTIME	93.248
FIRST SYSTEM FAILURE OCCURRED AT	955.634
NUMBER OF SYSTEM FAILURES WAS	16

Figure 8. Interim Results of Figure 3 Simulation

3.5 Final Results

The final output of the RBD simulation model is a measure of four reliability indicators for the system as a whole. The four measures that have been selected are: System Availability, First Failure Time, Mean Time Between Failure (MTBF), and Mean Time To Repair (MTTR). The results from each simulation run is maintained for each of these variables and a statistical analysis performed to yield the mean, standard deviation, and a 90% Confidence Interval based upon the empirical results. Additionally, the percentile distribution is shown in 5% increments.

Figure 9 gives the final output for the Figure 3 simulation. These results show considerable variance for all four variables. At a 90% observed confidence level, availability varies between 87% and 96% with a mean of 92.2%. First failure time varies widely between 48 and 1758 minutes of simulation time. MTBF and MTTR show simular variability. Consequently, variance information must closely examined in analyzing effectiveness measures in reliability.

4. EXAMPLE OF COMPLEX SYSTEM

Figure 10 depicts a more complex system, that is not readily analyzed. Structurally, it cannot be broken down into simple series and parallel components. Mean failure probabilities can be calculated by several deterministic means (such as the event space method, path tracing, decomposition, and cut and tie sets).

FINAL RESULTS OF RELIABILITY SIMULATION

	AVAILABILITY PERCENT	FIRST FAILURE	MTBF	MTTR
MEAN	92.246	691.125	705.090	54.866
SD	2.297	604.330	208.670	8.224
PERCENTILE	RANKINGS			
1	85.143	9.931	388.787	34.953
5	87.127	47.549	428.302	41.567
10	89.454	67.865	464.198	44.851
15	90.196	95.642	508.017	46.527
20	90.673	131.854	535.007	48.467
25	90.991	163.165	554.394	49.768
30	91.245	209.882	571.911	50.364
35	91.603	349.098	591.920	50.784
40	91.989	394.143	609.939	51.894
45	92.134	478.108	636.873	52.774
50	92.480	536.559	653.157	53.528
55	92.601	612.642	690.657	54.981
60	93.108	682.243	711.006	56.308
65	93.321	804.360	748.729	57.426
70	93.646	884.099	776.595	58.424
75	93.765	955.634	802.463	60.162
80	93.968	1114.014	840.843	61.463
85	94.200	1365.268	950.135	63.132
90	94.498	1576.730	1049.327	65.792
95	95.932	1757.512	1079.697	69.248
100	97.903	3075.308	1354.144	75.404

90% CONFIDENCE INTERVALS		
AVAILABILITY	87.127 TO	95.932
FIRST FAILURE	47.549 TO	1757.512
MTBF	428.302 TO	1079.697
MTTR	41.567 TO	69.248

Figure 9. Final Results of Figure 3 Simulation

However, no variance information can be found by these methods, nor can other pertinent information such as availability, first failure time, and MTTR. These answer are best found by simulation.

In Figure 10 and corresponding input file in Figure 11, block 1 (the starting block) and block 14 (the termination) do not represent real components in the system. Block 14's output status however represents the status of the overall system. Blocks 2 and 3 are not shown in the figure, but are the dummy blocks used to connect block 1 with 5 outputs. Blocks 4 through 8 are identical are characterized by exponential failure and normal repair. Blocks 9 through 11 are identical and are characterized by uniform failure and log normal repair. Similarly blocks 12 and 13 are identical with exponential failure and normal repair. Blocks 10, 12, 13 and 14 have input thresholds of 2 which once again complicates any deterministic evaluatuation. The results of the simulation are displayed in the Figure 12. It is evident that this system does exhibit considerable variability in the key variables being traced, as was the case in the simple 2 of 3 system.

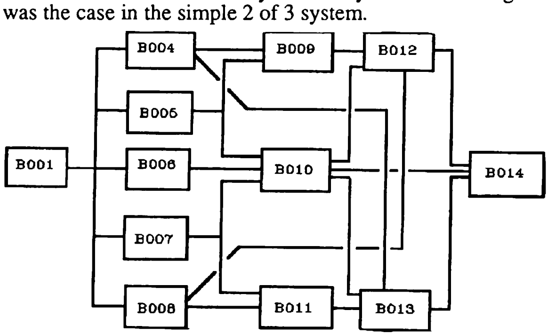

Figure 10. Complex System Simulation

```
100000.0        FALSE    FALSE
89655
END
B001   2   3
B002   4   5   6   7
B003   8
B004   9  13
B005   9  10
B006  10
B007  10  11
B008  11  12
B009  12
B010  12  13  14   0   2
B011  13
B012  14   0   0   0   2
B013  14   0   0   0   2
B014   0   0   0   0   2
END
F004   2    500.
R004   3    100.      25.
F005   2    500.
R005   3    100.      25.
F006   2    500.
R006   3    100.      25.
F007   2    500.
R007   3    100.      25.
F008   2    500.
R008   3    100.      25.
F009   4   1200.     100.
R009   5    300.      75.
F010   4   1200.     100.
R010   5    300.      75.
F011   4   1200.     100.
R011   5    300.      75.
F012   2   1500.
R012   3    450.     105.
F013   2   1500.
R013   3    450.     105.
END
```

Figure 11. Input File for Figure 4

RELIABILITY SIMULATION

	AVAILABILITY PERCENT	FIRST FAILURE	MTBF	MTTR
MEAN	97.423	3616.930	2736.558	64.627
SD	1.657	3176.569	710.801	22.204
PERCENTILE RANKINGS				
1	90.715	125.377	1224.456	35.451
5	93.182	230.855	1468.757	42.564
10	95.182	468.889	1830.601	45.705
15	96.905	740.433	2075.556	48.788
20	97.219	1259.440	2173.599	49.975
25	97.289	1633.920	2226.174	51.859
30	97.494	1861.347	2363.981	53.358
35	97.547	2064.513	2479.419	54.919
40	97.659	2226.202	2554.315	55.641
45	97.724	2517.091	2601.525	56.509
50	97.828	2949.806	2697.478	58.176
55	97.963	3260.363	2744.716	59.824
60	98.046	3647.910	2828.388	62.276
65	98.143	3878.934	2894.816	64.752
70	98.186	4631.680	2998.030	66.400
75	98.301	5282.057	3101.869	70.067
80	98.425	5632.555	3239.502	72.513
85	98.549	5801.035	3444.800	75.335
90	98.625	6453.858	3646.291	95.435
95	98.740	7705.967	3982.457	109.090
100	99.186	22223.243	5255.399	188.108

90% CONFIDENCE INTERVALS			
AVAILABILITY	93.182	TO	98.740
FIRST FAILURE	230.855	TO	7705.967
MTBF	1468.757	TO	3982.457
MTTR	42.564	TO	109.090

Figure 12. Final Results of Figure 4 Simulation

5. CONCLUDING REMARKS

The structure of reliability models makes them readily solvable by general programs without individual customization. The simple approach to connecting blocks demonstrated in this paper can be used to develop powerful general purpose simulation models that need only the system block diagram and component reliabilities as inputs. Simulation is important in the analysis of complex systems as often no definitive analytical solutions are available. The power of simulation models to analytically develop variance information for key variables associated with the overall system is a significant advantage over deterministic solutions.

REFERENCE

Kapur K.C. and L.R. Lamberson (1977), *Reliability in Engineering Design*, John Wiley, NY.