Evaluating computer systems through simulation

By L. Rowell Huesmann* and Robert P. Goldberg†

A review of computer systems simulation is presented. Existing and proposed programs for simulating computer systems are described and recommendations are made about the desirable characteristics of a simulator. Attention is primarily focused upon those simulators that may be used to study time-sharing, multiprogramming, and multiprocessing systems. It appears that a higher order, special purpose simulation language is needed for simulating computer systems. CSS and LOMUSS II seem to be representative of the most promising type of computer system simulators.

As computer systems have grown in size and computer implementations have grown in sophistication, it has become more and more important to develop means for evaluating different hardware and software configurations both prior to and after installation. One approach to such evaluation has been through digital computer simulation of proposed computer systems.

To use simulation techniques in evaluating different computer systems, one must be able to specify formally the expected job mix and constraints under which the simulated system must operate, e.g., operating time per week. Equally important, one must carefully select a set of characteristics on which the competing systems will be judged. For different installations the most important characteristics may well be different. Each system under consideration is modelled, simulation runs are executed, and the results are compared on the selected characteristics.

Unfortunately, the ideal case seldom occurs. Often the available information about the computer system's expected job mix is very limited. Furthermore, it is a well-known fact that an installation's job mix itself may be strongly influenced both qualitatively and quantitatively by the proposed changes in the system. For example, many of the difficulties with early time-sharing systems can be attributed to the changes in user practices caused by the introduction of the system.[16] When statistics on job mix are available, they are often expressed in averages. Yet, it may be most important to simulate a system's performance under extreme conditions. Finally, it is often difficult to show that a simulation is valid—that is, that it actually does simulate the system in question.

Although the simulation of computer systems on digital computers is a difficult task, it appears to be an increasingly popular method for evaluating computer systems. Perhaps the foremost reason for simulation's popularity is the lack of a viable alternative. Several recent papers[9, 42] have proposed the use of Markov models as one alternative. Each computer system is represented as a Markov system, and its steady state is used to calculate performance statistics. Unfortunately, the specification of each system as a Markov chain and the estimation of the transition probabilities are tedious tasks which may be more expensive than several simulation runs. Aside from Markovian methods, there does not seem to be any formal method of modelling computer systems which can produce results comparable to simulation.

The remainder of this paper will review some of the more promising efforts to build simulators for time-sharing, multiprogramming, and multiprocessor systems, and attempt to extract from these programs the characteristics of an ideal computer simulator.

Type of input

One very important characteristic useful in classifying simulators is the type of input which the simulator requires. In other words, how must the computer system, its operating constraints, and its job mix be specified? Ideally, one would want a completely parameterized program capable of simulating any existing or proposed computer system. The only input needed to describe the system would be simple parameter values. As yet, no such program has been proposed. These authors feel that the development of such a program in the near future is not feasible because of the differences between hardware and operating systems to be simulated, the rapid changes in computer technology and organization, and the wide variations in simulation objectives.

One special-purpose parameterized program which is widely used for evaluating computer systems is SCERT (System and Computer Evaluation and Review Technique).[6, 17] SCERT is not a simulation program in the usual sense. It does not have a clock which it advances to simulate real time. Instead it uses a "table-look-up" and a series of empirically determined equations to estimate a computer system's behaviour under a given job mix. Written in assembly language, SCERT allows the user to specify particular hardware configurations by reading in the manufacturer's model numbers. This eliminates the time-consuming work needed to specify

* *Carnegie Institute of Technology, Pittsburgh, Pennsylvania.*
† *M.I.T. Lincoln Laboratory, Lexington, Massachusetts.* (Research for this paper was performed when the authors were employed by The MITRE Corporation, Bedford, Mass.)

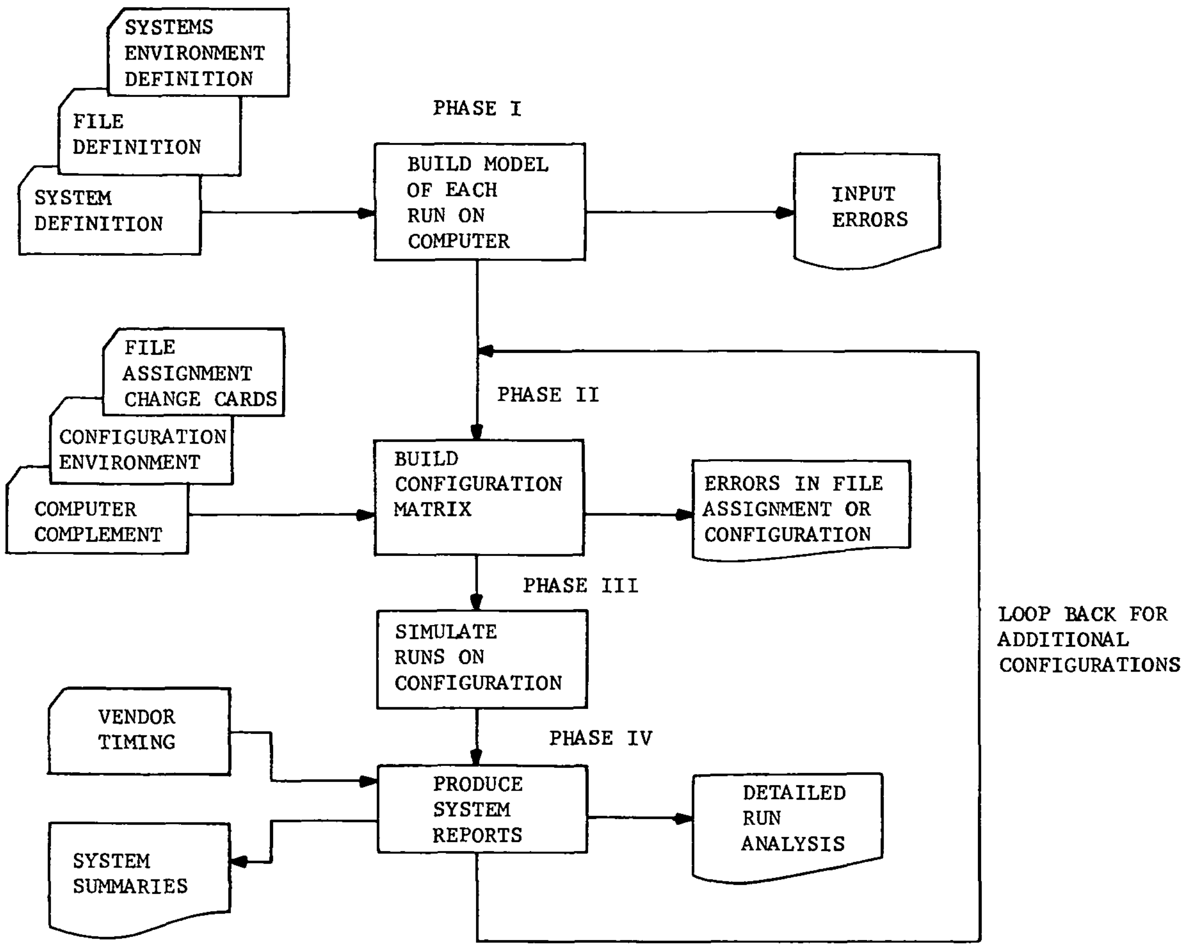
SYSTEMS ENVIRONMENT DEFINITION
FILE DEFINITION
SYSTEM DEFINITION
PHASE I
BUILD MODEL OF EACH RUN ON COMPUTER
INPUT ERRORS
FILE ASSIGNMENT CHANGE CARDS
CONFIGURATION ENVIRONMENT
COMPUTER COMPLEMENT
PHASE II
BUILD CONFIGURATION MATRIX
ERRORS IN FILE ASSIGNMENT OR CONFIGURATION
PHASE III
SIMULATE RUNS ON CONFIGURATION
LOOP BACK FOR ADDITIONAL CONFIGURATIONS
VENDOR TIMING
PHASE IV
PRODUCE SYSTEM REPORTS
DETAILED RUN ANALYSIS
SYSTEM SUMMARIES

Fig. 1. SCERT

the relevant characteristics of the simulated computer. However, there are two main disadvantages to such an approach. First, one cannot evaluate any piece of equipment which is not included in the SCERT tables; and second, one cannot get a real feel for what characteristics are the limiting ones for a particular hardware configuration. SCERT has very few input parameters relating to operating system functions and very limited capabilities for evaluating time-sharing, multiprogramming, and multiprocessing systems. Nevertheless, the use of SCERT has demonstrated the value of a simulation-like approach toward hardware and operating system evaluation. (See **Fig. 1**.)

Many other simulation programs also allow a few, but not all, of the system's characteristics to be specified by parameters. However, most of the important characteristics of the simulation, e.g., computer configuration and operating system, must be described by routines written in a programming language. A few simulation programs have required that these descriptive routines be written in the simulator's own assembly or algebraic compiler language.

Fine and McIsaac[10, 11] and Jones[25] have developed several different simulators of this last type for evaluating computer system configurations. Designed to simulate variations of the SDC time-sharing system, these programs require that many of the hardware characteristics be specified in JOVIAL or Q-32 assembly language. Many operating system characteristics and some hardware characteristics, however, are parameterized and can be easily varied. These parameterized factors include core storage size, drum and disc size, drum and disc access speed and transfer rate, system malfunction rate, malfunction recovery time, quantum time, queueing discipline, and number of remote channels. Although these simulation programs can handle time-sharing and multiprogramming, all of them assume that the configuration being simulated has only one CPU. The job mix for the simulated system is specified statistically. The actual probability distributions are read in for each of thirteen characteristics (e.g., I/O calls, program length). The simulator then generates jobs according to these distributions. As output, the simulation programs produce complete records of response times, overhead times, swap times, and other similar quantities. All of SDC's simulation programs work on a "next event" basis—that is, the simulated time clock is advanced to the time of the next event. Typically, they require about

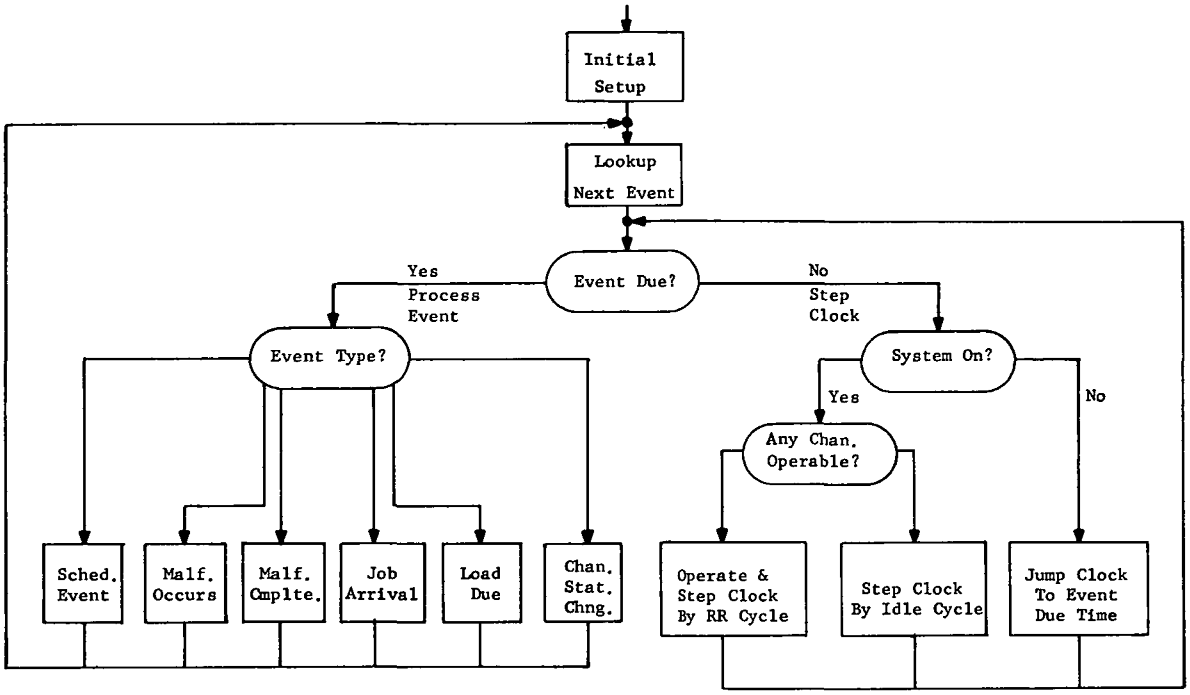

Fig. 2. SDC simulation *(Courtesy of SDC)*

30 minutes to simulate 24 hours when run on the Q-32 computer. (See **Fig. 2.**)

A similar approach toward simulating large time-sharing, multiprogramming systems has been taken by Neilson[37] in his FORTRAN IV simulation of IBM's 360/67.

CTSS

Probably the most sophisticated computer simulator developed to accept computer system and job mix descriptions in an algebraic language is Scherr's CTSS simulator.[41] His simulator consists of a series of special statements and routines through which the computer system is described, and a processor which conducts the actual simulation. To specify the hardware and operating system characteristics of the simulated system, one constructs a program from Scherr's special simulation statements, MAD subroutines which Scherr provides, and MAD in general. This method has the advantage that any operating system or hardware configuration can be specified in as much or as little detail as is desired. Of course, if the hardware and operating systems to be simulated are extensive, the programming effort needed to specify them will be correspondingly tedious and difficult. The job mix for the simulated system is specified by writing a program which generates I/O calls, execution times, etc., according to desired statistics. Scherr provides MAD subroutines and special statements which make this process fairly easy. Scherr's program records statistics on response time, throughput, queues, overhead, and other factors during the simulation. Three factors distinguish Scherr's approach. First, since the simulator system and the operating system to be simulated are written in the same language, existing operating system subroutines can be used as an integral part of the simulator. Second, Scherr added statistics gathering routines to the CTSS supervisor in order to obtain accurate data on the job mix. Third, Scherr's simulator has produced output very close to the CTSS system. (See **Fig. 3.**)

Special languages

Those programs which require the simulated computer system and job mix to be specified in algebraic or assembly languages have proved useful; but as general computer systems simulation tools, they require too much difficult recoding to be completely satisfactory. One way to improve upon this situation has been to use languages specifically designed to simulate systems. Teichroew and Lubin[44] in a recent review have listed more than twenty languages, among them GPSS,[24] SIMSCRIPT,[34, 35] SOL,[30] and CSL.[4] These simulation languages allow the modeller to specify the computer configuration and job mix in a more convenient manner.

GPSS has been used by several groups to simulate time-sharing computer systems. Campbell, McCabe, and Nevans[5] have used a GPSS simulation to evaluate different hardware in a large time-shared information retrieval system. Stanley and Hertel[43] have used GPSS to simulate the NASA Manned Spacecraft Center's Real Time Computing Complex. This simulator is of a more general nature than any of the previously mentioned programs. Within the System 360 limits, different hardware devices can be specified by inputting different

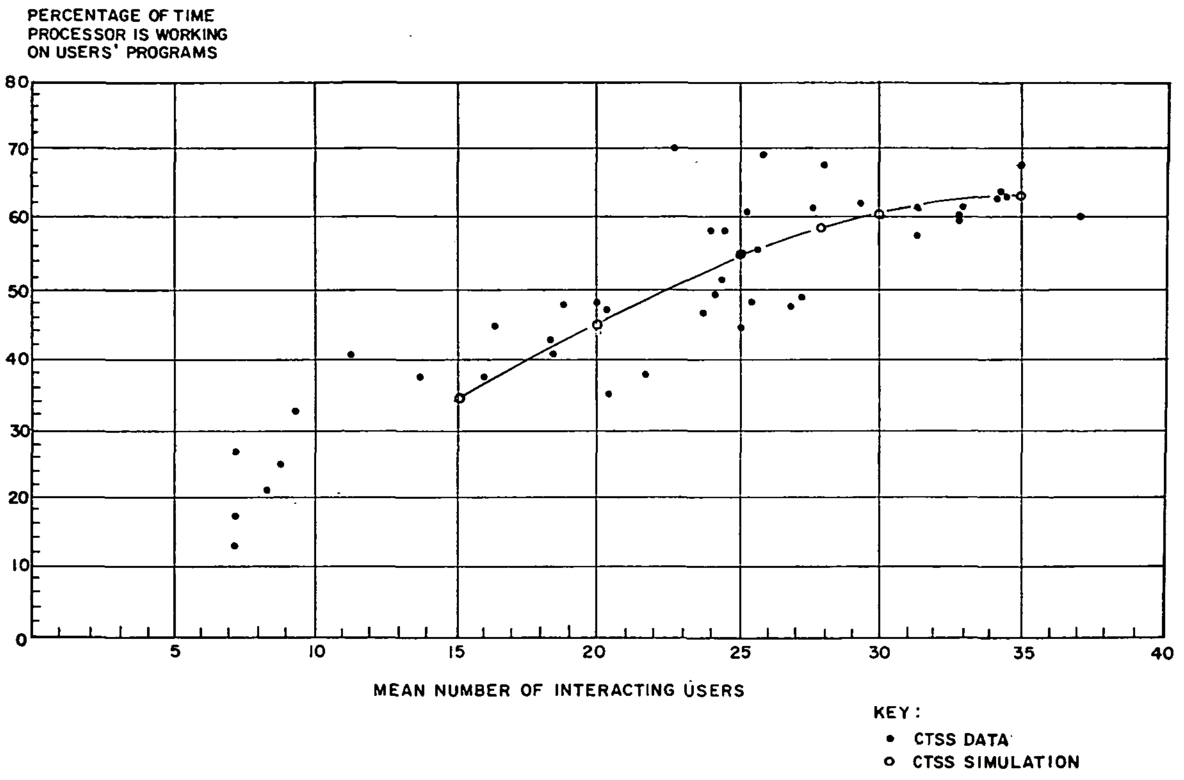

Fig. 3. Comparison of CTSS data and simulation *(Courtesy of M.I.T. Press)*

parameter values. The Real Time Operating System being simulated is described by routines written in GPSS. Similarly, the job mix is specified by GPSS routines. One other important part of the simulation effort is a statistics gathering program. This program is used to record statistics on the job mix which a computer system is handling. With only a small degradation in systems performance this addition to the monitor provides unusually accurate data on the systems job mix and thus produces input for a more accurate simulation.

SIMSCRIPT and SIMTRAN[2] have been extensively used to simulate time-sharing, multiprogramming and multiprocessing systems. Katz[26] has simulated a multiprocessing system with SIMSCRIPT. He specifies most of the characteristics of a "direct-coupled" 7090–7040 system with routines in SIMSCRIPT. However, buffer size, the number of input and output stations, some of the operating system characteristics, and some of the job mix characteristics are parameterized. For example, by varying parameters one can compare fixed and dynamic scheduling strategies, different queueing disciplines, and different job cutoff parameters. Each job to be simulated is specified in terms of its input from each peripheral device, its output to each peripheral device, its rate of I/O calls, and its estimated execution time under a no-overhead assumption. This last parameter, in fact, permits one to examine the effect of varying CPU speed. The time at which each job enters the system, the remote device for each job, and each job's priority must also be given. To simulate about one hour of work with Katz's program requires about 20 seconds of IBM 7094 time. This low ratio of simulator execution time to simulated time is achieved by a system state approach in SIMSCRIPT where time is always advanced to the next point where the computer system will change its state. (See **Fig. 4.**)

Another simulation program designed to simulate multiprocessor systems is being developed by R. Goldstein at Lawrence Radiation Laboratory.[13] Written in SIMTRAN, this program is specifically designed to simulate the OCTOPUS computer system at LRL which includes an IBM 7030 STRETCH, two IBM 7094's, two CDC 6600's, one CDC 3600, two PDP 6's, an IBM 1401, and various I/O devices. A parameterized input table specifies the general multiprocessing configuration, the data transmission rates, memory sizes and buffer sizes. Any other hardware variations and operating system characteristics are introduced with SIMTRAN routines. As output the program produces figures on actual memory utilization, graphs of memory access time, graphs of overhead, graphs of response time, and graphs of several other relevant variables.

The general purpose simulation languages appear to make the construction of successful simulators for computer systems easier. However, none of the specific programs written in these languages are sufficiently parameterized to allow the evaluation of a wide range of computer systems. Furthermore, constructing new routines to describe each new simulated computer system

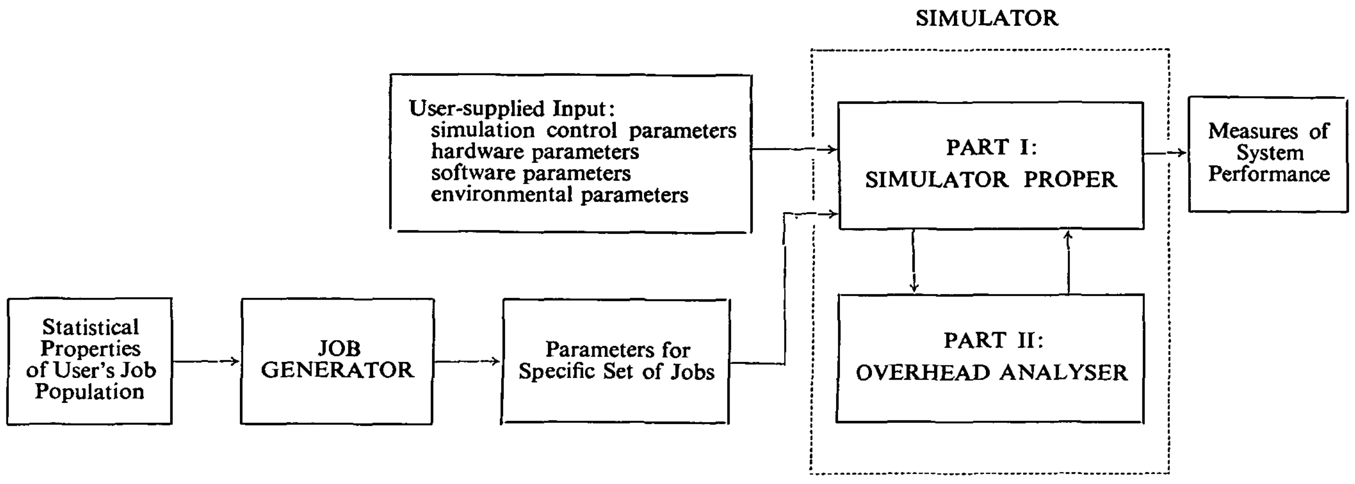
SIMULATOR
User-supplied Input: simulation control parameters hardware parameters software parameters environmental parameters
PART I: SIMULATOR PROPER
Measures of System Performance
Statistical Properties of User's Job Population
JOB GENERATOR
Parameters for Specific Set of Jobs
PART II: OVERHEAD ANALYSER

Fig. 4. Katz's DCS simulator

(Courtesy of Spartan Books)

does not seem simple enough with these general simulation languages to make their use economically feasible. Kiviat[27] has suggested that one solution to this dilemma is to build higher order, more specialized, and easier to use simulation languages out of the existing ones. Two of the recently proposed simulation languages, SIMSCRIPT II and IBM's "New Simulation System"[31, 38] have provisions for adding MACROS for special uses. These MACROS or higher level languages could be used specifically to describe computer systems.

Higher level languages

The development of higher level languages for describing computer systems for simulation purposes has been carried out in at least two instances.

IBM has been using a dynamic simulator called CSS (Computer Systems Simulator[23]) to evaluate hardware and operating system configurations. CSS is capable of simulating hardware systems in almost any type of time-sharing or multiprogramming mode. While the hardware configuration is described by means of "equipment specification cards" which simply list timing information on the units, the operating system and job mix are described by a program in the CSS modelling language. This language contains only a small number of instructions, but they are specifically designed to facilitate the specification of a computer program statistically or deterministically, and vaguely or in detail. During the simulation the CSS program automatically records statistics on response time, overhead, and several other important factors. Furthermore, the user can request data on any special factors that interest him.

The second example is the Lockheed Multipurpose Simulation System (LOMUSS II[20, 33]). Written in SIMSCRIPT, this program treats the operation of a multiprocessing computer system as a problem of allocating resources to demanding jobs. (See **Fig. 5.**) Any hardware or operating system characteristics desired can be specified by means of "operation definitions". For example, the CPU is specified by a list of symbols representing its relevant operations. Paired with each operation is the time required to perform that operation. In LOMUSS II terminology, the CPU is an all-or-none resource, and the time required to perform an operation represents the time a job must occupy that resource while performing the operation. A disc, however, is specified not only by a list of its relevant operations and the rate at which they can be performed (e.g., transfer rate) but also by its size. In LOMUSS II, memory size is considered to be a continuous resource, and any amount can be used by any job. Operation symbols which are not already defined in LOMUSS II can be created and defined by the user. The definition takes the form of a routine written in a language called MDL (Model Description Language). This, like CSS's modelling language, is designed to facilitate such definitions. Thus, if the user wishes to specify an unusual disc accessing procedure, he can define his own procedure easily and conveniently. Although some characteristics of the operating system are parameterized, most are specified by routines in MDL. In these cases, the user need only input a number. This completely general method of computer configuration specification requires some research and work on the LOMUSS II user's part, but it requires nowhere near the amount of work that writing a simulation program in GPSS or SIMSCRIPT would require. LOMUSS II appears to

DISCRETE RESOURCES

Resource	*Type Code*	*Operation*	*Rate*
Main Frame	06	61	16×10^{-6} (Add)
		65	64×10^{-6} (Multiply)
Tape	11	111	32000 (Transfer Bytes)
		112	35 (Rewind Speed)

CONTINUOUS RESOURCES

Resource	*Type Code*	*Size*
Memory	120	32768
Drum	130	256

Fig. 5. Sample LOMUSS II configuration definition

allow the user to specify any configuration whether it is a common one or not; it allows him to determine exactly what characteristic within any part of the system is a limiting characteristic; and it permits him to specify different parts of the system in differing depths of detail depending on his needs. Similarly, the method for specifying a system's job mix for LOMUSS II also seems to permit a wide variation in level of detail. Each job type is described by a small program written in MDL. The description can be as general as only specifying an approximate execution time, or it can be as specific as specifying the number of ADD's and MULTIPLY's in a program. Furthermore, MDL allows probabilistic variations to be introduced into these job descriptions. The supervisory program which determines the arrival time of the different job types is also written in MDL. The output which a LOMUSS II simulation produces varies according to the user's needs. Generally, two types of records are maintained during the simulation: a record of the state of each resource at different points in time, and a profile of each job. Thus, statistics on response time, processor idle time, memory utilization, throughput, queue behaviour, and many other factors are available. The time for a LOMUSS II simulation averages about one-third of the real time period being simulated.

Ideal characteristics

These last two approaches toward constructing a generally applicable program or language for simulating computer systems appear to be a step above any of the previous attempts. The analysis and comparison of all these various approaches to computer systems simulation enables one to list a set of ideal characteristics for a computer systems simulator. First, one should be able to simulate a wide variety of hardware and operating system configurations without extensive recoding for each application. When reprogramming is necessary, it should be in a special language for modelling computer systems. Furthermore, the program should be able to simulate different parts of the computer configuration at different levels of specificity. One person might want to perform a simulation just for the purpose of examining a disc accessing strategy, while another person might be interested in how a system's response varied with the number of users. Also, by permitting different levels of simulation in different parts of the configuration, one avoids the unnecessary detailed descriptions which would be required if a fine level of simulation were used for the entire computer configuration. Similarly, the program should allow one to specify the job mix either very generally in a probabilistic manner or very specifically, even to the point of giving the number of ADD's and MULTIPLY's in each program. For widest applicability, a computer system simulator should be a dynamic simulator—that is, it should have a clock whose advances represent real-time advances. As output, the simulator should produce snapshots of the state of the computer system at different time periods, and profiles of each job that goes through the system. Also, it should record statistics on such factors as response time, queue behaviour, throughput, processor idle time, memory utilization, I/O device utilization, and monitor efficiency.[18] Three other characteristics which are important in a computer system simulator are proven validity, high reliability (producing about the same result on different runs), and efficiency (a high ratio of simulated real-time to execution time of the simulator). Finally, associated with a computer systems simulator at any installation should be a statistics-gathering program which, as part of that computer system's monitor, can obtain accurate statistics on that installation's job mix.

Thus systems simulation has proved successful in studying many specific computer systems. If used as indicated in this paper, computer systems simulation can be a useful general-purpose tool for evaluating proposed hardware and software configurations, and for controlling the orderly evolution of an installation's computing facility.

Acknowledgements

The authors wish to thank Messrs. Jack D. Porter and John H. McIntosh for their encouragement and assistance in the preparation of this paper.

Bibliography

1. Blunden, G. P., and Krasnow, H. S. "The Process Concept as a Basis for Simulating Modeling", IBM Corporation, IBM-ASD-TR-17-181, Yorktown Heights, N.Y., 1965.
2. Braddock, D. M., C. B. Dowling, and K. Rochelson. "SIMTRAN Manual", IBM Corporation, Poughkeepsie, N.Y., July 1965.
3. Brown, J. W., W. A. Grant, L. R. Kimble, G. Langnas, and T. G. Sanborn. "Programmer's Manual for SIMCON", Space Technology Laboratories, Inc., Redondo Beach, Cal., July 1960.
4. Buxton, J. H., and J. G. Laski. "Control and Simulation Language", *The Computer Journal*, 5(3), 1963, 194.
5. Campbell, J. B., J. R. McCabe, and E. S. Nevans. "The Application of Large Scale Computers to U.S. Air Force Information Systems", Electronic Systems Division, AFSC, ESD-TR-66-137, L. G. Hanscom Field, Bedford, Mass., March 1966.
6. COMRESS Incorporated. "SCERT Users Manual", Washington, D.C., March 1965.
7. Denning, J. P. "Queueing Models for File Memory Operation", M.I.T., Project MAC, MAC-TR-21, Cambridge, Mass., October 1965.
8. Famolari, E. "FORSIM IV, FORTRAN IV Simulation Language User's Guide", The MITRE Corporation, SR-99, Bedford, Mass., January 1964.

9. Fife, D. W. "An Optimization Model for Time-Sharing", *AFIPS*, 28, Spring 1966, 97.
10. Fine, G. H., and P. V. McIsaac. "Simulation of a Time Sharing System", System Development Corporation, SDC-SP-1909, Santa Monica, Cal., December 1964.
11. Fine, G. H. "Preliminary Investigations in Time-Sharing Simulation", System Development Corporation, SDC-TM-2203, Santa Monica, Cal., January 1965.
12. Gibson, C. T. "Time-Sharing in the IBM System/360 Model 67", *AFIPS*, 28, Spring 1966, 61.
13. Goldstein, R. (Personal Communication.) Lawrence Radiation Laboratory, Livermore, Cal., July 1966.
14. Gordon, G., and K. Blake. "Systems Simulation with Digital Computers", *IBM Systems Journal*, 3(1), 1964, 14.
15. Graham, K. R. "A Comparison of the Simplex and Semi-Duplex Configuration for the IBM 360/67", Unpublished Paper, Carnegie Institute of Technology Computation Center, Pittsburgh, Pa., Spring 1966.
16. Green, B. F., and L. R. Huesmann. "Computer Time-Sharing", Unpublished Report, Carnegie Institute of Technology, Pittsburgh, Pa., 1965.
17. Herman, D. J., and F. C. Ihrer. "The Use of a Computer to Evaluate Computers", *AFIPS*, 25, Spring 1964, 383.
18. Huesmann, L. R., and R. P. Goldberg. "A Review of Computer Systems Simulation", The MITRE Corporation, MTR-258, Bedford, Mass., August 1966.
19. Hutchinson, G. K. "Generalized Models of Information Processing Systems", Paper presented at Operations Research Society of America National Meeting, May 1966.
20. Hutchinson, G. K. "A Computer Center Simulation Project", *C. ACM*, 8(9), September 1965, 559.
21. Hutchinson, G. K., and J. N. Maguire. "Computer Systems Design and Analysis Through Simulation", *AFIPS*, 27, Fall 1965, 161.
22. IBM Corporation. "Analysis of Some Queueing Models in Real-Time Systems", IBM-F20-0007-0, White Plains, N.Y., 1964.
23. IBM Corporation. "Computer System Simulator: Application Descriptions", IBM-F20-0322-0, White Plains, N.Y., 1964.
24. IBM Corporation. "General Purpose Systems Simulator III Introduction", IBM-B20-0001-0, White Plains, N.Y., 1965.
25. Jones, B. Personal Communication, System Development Corporation, Santa Monica, California, June 1966.
26. Katz, J. H. "Simulation of a Multiprocessor Computer System", *AFIPS*, 28, Spring 1966, 127.
27. Kiviat, P. J. "Development of New Digital Simulation Languages", The RAND Corporation, Santa Monica, Cal., 1966.
28. Kleinrock, L. "Sequential Processing Machines Analyzed with a Queueing Theory Model", *J. ACM*, 13(2), April 1966, 179.
29. Knuth, D. E., and J. L. McNeley. "A Normal Definition of SOL", *The IEEE Transactions on Electronic Computers*, August 1964.
30. Knuth, D. E., and J. L. McNeley, "SOL—A Symbolic Language for General Purpose Systems Simulation", *The IEEE Transactions on Electronic Computers*, August 1964.
31. Krasnow, H. S. "Highlights of a Dynamic System Description Language", IBM Corporation, IBM-ASD-TR-17-195, Yorktown Heights, N.Y., 1966.
32. Krasnow, H. S., and R. A. Merikallio. "The Past, Present, and Future of General Simulation Languages", *Management Science*, 2(2), 1964, 236.
33. Lockheed Missiles and Space Company. "Lockheed Multipurpose Simulation System (LOMUSS II)", Sunnyvale, California, June 1966.
34. Markowitz, H. M., and B. Dimsdale. "A Description of the SIMSCRIPT Language", *IBM Systems Journal*, 3(1), 1964, 57.
35. Markowitz, H. M., B. Hauser, and H. W. Karr, (RAND), *SIMSCRIPT: A Simulation Programming Language*, Englewood Cliffs, N.J., Prentice Hall, 1963.
36. McIntosh, J. H. "COMET Internal Timing Analysis". The MITRE Corporation, Memo MD84–941, Bedford, Mass., June 1966.
37. Neilson, N. Personal Communication, Stanford University, Palo Alto, Cal., June 1966.
38. Parente, R. J. "A Language for Dynamic System Description", IBM Corporation, IBM-ASD-TR-17-180, Yorktown Heights, N.Y., 1965.
39. Porter, J. D. "Multiprogramming/Multiprocessing Evaluation Studies", The MITRE Corporation, Bedford, Mass., Paper presented at Inter-Service Conference on EDPE Selection Methodology and Data Exchange, July 1966.
40. Rosen, S. B., E. S. Nevans, R. M. Walken, W. Asmuth, and L. C. Marshall. "A Study of System Modeling Techniques", Electronic Systems Division, AFSC, ESD-TDR-63-612, Hanscom Field, Bedford, Mass., October 1963.
41. Scherr, A. L. "An Analysis of Time Shared Computer Systems", M.I.T., Project MAC, MAC-TR-18, Cambridge, Mass., June 1965.
42. Smith, J. L. "An Analysis of Time-Sharing Computer Systems Using Markov Models", *AFIPS*, 28, Spring 1966, 87.
43. Stanley, W. I., and H. F. Hertel. "The Performance Measurement and Analysis of System/360 Multiprogrammed Systems", IBM Federal Systems Division, Houston, Texas, July 1966.
44. Teichroew, D., and J. F. Lubin. "Computer Simulation-Discussion of the Technique and Comparison of Languages", *C. ACM*, 9(10), October 1966, 723.
45. Youchah, M. I., and D. D. Rudie. "A Universal DPC Simulator Applied to SACCS Program System Design", System Development Corporation, SDC-SP-924, Santa Monica, Cal., July 1963.
46. Youchah, M. I., D. D. Rudie, and E. I. Johnson, (SDC). "The Data Processing System Simulator", *AFIPS*, 26, Fall 1964, 251.