The author assumes that the reader is familiar with the content of the preceding paper, "A General Purpose Systems Simulator," by G. Gordon. Two dissimilar examples are provided to illustrate various aspects of simulation in the systems engineering process. One example involves the study of an IBM 7040—IBM 7090 *computer complex for scientific applications. The other concerns an* IBM 1410 *Tele-processing system for a stock brokerage house.* □ *In addition to illustrating the paper, both examples are of intrinsic interest. The first presents a new philosophy of multiprocessing. The second examines a method of integrating communication facilities with an information processor.*

Simulation in systems engineering

by E. C. Smith, Jr.

Systems engineering is concerned with the synthesis and analysis of systems of men and machines having specified objectives. Such performance objectives are commonly expressed in terms of human values and, as a consequence, are often necessarily subjective. Furthermore, the systems under consideration invariably exploit the interaction of men with physical components. It is this high degree of human involvement which particularly distinguishes systems engineering from the more conventional fields of engineering. Thus, the design and analysis of a message handling information system is a systems engineering activity, whereas the design and evaluation of a water pump falls within conventional practice.

The conduct of systems engineering invariably includes the conscious use of some form of modeling: either physical, such as a model airfoil for a wind tunnel; or conceptive, by formal mathematics or logical simulation. The General Purpose Systems Simulator[1] is one instrument which the systems engineer may use to construct and manipulate a conceptive model. It can be used for models of quite diverse and dissimilar physical systems. However, it is far from completely general. Even if attention is restricted to information processing systems, the simulator is inadequate for the expression of many conceptual models.

Use of the simulator can be appreciated only in the terms of its role in the total systems engineering process. The purpose of this paper is to help develop insight into its proper use through discussion of two systems engineering studies; one relating to a

computer complex for scientific applications, and the other to a message handling system for a brokerage firm.

Relationships between system components are the raw material of the systems engineer. The systems engineer must first determine the performance criteria—the values against which the desired system will be measured. He must then select the system's components and their interrelationships so that the resultant system will measure best against this criteria. Thus, once the performance criteria have been established, his job is one of selection and evaluation. A simulation model may be used in the selection of the system's components and their relationships, or it may be used to help evaluate a specific system configuration, or both. These two phases may proceed concurrently, occasionally without conscious distinction between them, or they may proceed independently. Indeed, different models may be used in these two phases.

Occasionally, selection of the system configuration may be dictated by equipment availability or economics so that only one system is possible. Then, the choice is whether or not to have the system. Modeling provides an evaluation which will permit this choice to be made with minimal risk.

steps in the use of simulation

Simulation is not replication. The simulation (or mathematical) model possesses at best a relatively small number of properties which correspond (*always in an approximate way*) to reality. In order to use simulation, the systems engineer must:

1 Determine those properties of reality in which he is interested.
2 Determine those properties of reality which could conceivably have a significant influence upon those in which he is interested.
3 Specify the relationships between the above properties.
4 Use a modeling tool (mathematical theory, General Purpose Systems Simulator language, etc.) to build a model.
5 Establish a univalent correspondence between reality and the identified entities and relationships of the model.
6 Manipulate the model.
7 Interpret the results of the manipulation.

Note that in a very real sense the first three steps build a verbal or conceptual model. The very act of writing down properties and their relationships necessitates simplification of the incredible complexity of reality—human language is bounded by the degree of complexity which man can grasp and understand. This type of modeling is very useful and is the kind of analysis that is often described as sitting back, thinking about the problem and seeing the kernel of it. Step 4 above might be rephrased as, "Use a modeling tool other than general language to build a more precise model."

model manipulation

Manipulation of the model produces the results. For a mathematical model expressed as a system of equations, manipulation consists of solving the equations. For a physical model or a model expressed in a language such as that of the General Purpose Systems Simulator, manipulation consists of the conduct

of experiments and the collection of statistics for later analysis. Manipulation in this case may also mean the varying of parameters in the model design and conducting more experiments to obtain further statistics for comparative purposes. (One might consider this the construction and manipulation of new, slightly different models, but it is more common to consider a model to include those variants obtained by changing parameters.) For a verbal model as developed in Steps 1, 2, and 3 above, the manipulation is more difficult to identify, but it consists primarily of the application of insight and mental analysis to the problem in order to draw conclusions.

The running of the General Purpose Systems Simulator program can produce many kinds of statistics about the model. For example, for a facility one may determine its average utilization (the fraction of the total elapsed time it is in use), the total number of transactions to use the facility, etc. For a queue, one may determine: the maximum length it attained, its average length, the number of times its length met certain pre-specified values, the total number of transactions which passed through the queue and the number of these which spent no waiting time at that point. Pages of statistics of this nature comprise the results of the model manipulation; the value of the entire procedure obviously depends upon the interpretation of these results. It is here that the evaluation in terms of human values is crucial. It is of the utmost importance to establish in advance the purpose of each particular simulation in the total systems engineering process, and to specify how that purpose can be achieved by interpretation of simulation results. Average queue length may be pertinent for one case, maximum length may be critical for another; average facility utilization may be a key factor for one case, maximum transaction life may be a key for another.

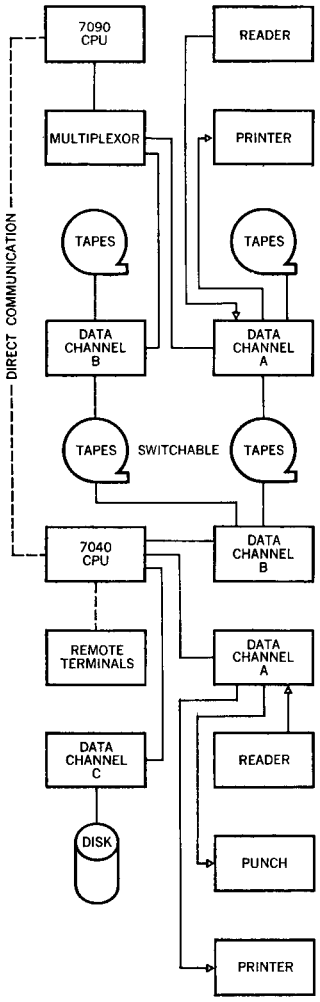
One of the questions most commonly asked of a model is its sensitivity to variations in its design or environment. For example, if a computer system is designed to handle a certain work load, what happens if the work load is increased slightly? The degradation of service may be slight, or it may be surprisingly great. Conceivably, a 10% increase in workload could cause the average job processing time to more than double because of the cumulative effect of several queues. The systems engineer often wants to vary model parameters in order to identify those to which the system is most sensitive. He can then devote most of his attention to those critical aspects of the total system.

First example : scientific computation

objectives

In order to better crystallize these ideas, consider the general data processing milieu of scientific computation. The objective of the use of data processing equipment in this context is to aid the engineer and scientist in the solution of his problems. Present day computing techniques are evidence of considerable progress in this area, but it is not difficult to propose much improvement. Rather than considering improvement by development of more

Figure 1



proposed multicomputer system

sophisticated problem statement languages, remote graphical terminals, etc., consider simply a reorganization of the central computer complex to handle problems originally stated in Fortran and/or FAP languages and presented to the complex from multiple sources. The objective is to serve the user better by:

1 Reducing the average turnaround time (time between the submission of programs to the complex and the presentation of the results).
2 Providing dynamic scheduling of programs for the central processor.
3 Providing dynamic scheduling of output devices (punches, printers, remote terminals).
4 Providing interrupt capabilities for on-line remote terminals.
5 Increasing central processor throughput (number of programs processed per unit of time).

The systems engineer should study each of the above in order to synthesize several alternative equipment configurations and operating systems which appear likely to accomplish the above. This would include a functional specification of supervisory monitor programs and operating procedures. Before attempting anything more elaborate than a cursory examination of the engineering difficulty and economic value of each of the proposed systems, each should be examined to determine if, indeed, it would accomplish each of the five points above. Stated simply, why perform a detailed economic analysis of a system that could not accomplish the task set forth?

We shall describe a simulation of one system which might be considered for this situation and set forth what the results of the simulation imply and, what may be more important, what they do *not* imply with regard to establishing the feasibility and desirability of the system. The reader should note that this is a hypothetical system and its discussion here is not necessarily intended to supply the optimal system for this situation.

The proposed system consists of two computers operating in concert to accomplish the scheduling and execution of programs. This computer complex, shown in Figure 1, consists of an IBM® 7090 and an IBM 7040 linked into an integrated system by programmed-switchable tapes for mass data transfers as well as other non-standard means of direct computer-to-computer communication. The IBM 7090 performs the compilation, assembly and execution of object programs. The IBM 7040 never relinquishes its control to an object program since it must constantly monitor the entire system, edit input, schedule and prepare input tapes for the IBM 7090, edit, schedule and initiate all output received by tape from the IBM 7090, and respond appropriately to remotely initiated messages transmitted by on-line Tele-processing® equipment.

Program information and data can arrive at the IBM 7040 simultaneously from several sources—card reader, tapes, and remote terminals. Such material is accumulated on the disk file

into complete programs. They are then, according to priority, written onto tape for transmittal to the IBM 7090 for processing. While passing through the IBM 7040 (to the disk or from the disk to tape), some preparatory processing is done. For example, Fortran and FAP statements are compressed by the elimination of null characters. Comments are stored on the disk with their programs, but are not put on the tape for the IBM 7090. Preliminary diagnostic editing is performed, possibly on a statement-by-statement validity basis rather than an inter-statement consistency level. Programs which fail such editing are appropriately handled by the IBM 7040 so that the problem originator receives the necessary diagnostic information to effect later corrections. Consequently, the IBM 7090 does not receive a program for assembly or compilation unless it has passed a minimum standard of error checking.

When creating an input tape for the IBM 7090, the IBM 7040 will select necessary library routines from the disk file and appropriately incorporate them with the input programs. Such routines, as well as previously assembled object programs, will be relocated by the IBM 7040 so that the IBM 7090 will read this information in absolute binary form for maximum efficiency in loading. Thus, the IBM 7090 throughput should be increased by a reduction in loading time and searching time for subroutines, and by an increase in the probability of successful compilations and assemblies effected by the pre-editing.

Periodically the IBM 7040 may monitor the execution of a program by the IBM 7090 by means of an interval timer and (currently unavailable) trapping and direct communication devices linking the machines. If the program execution is found to deviate from some previously specified pattern, the IBM 7040 should cause the IBM 7090 to terminate the processing of its current program. The IBM 7040 should perform preliminary diagnostic operations, condition the IBM 7090 to perform a more complete diagnostic function and then begin the next program. Consequently, the IBM 7040 always has complete control of the situation and can automatically restart the IBM 7090 even if an object program has erroneously put nonsensical information into virtually all of memory and entered an endless loop. For this reason this multiprocessor concept is often called a master-slave mode of operation.

purpose of simulating system

Since a primary requirement is to help the IBM 7090 to operate at maximum efficiency, the first question it is necessary to settle is whether or not the IBM 7090 would ever be kept waiting for work because the IBM 7040 could not present it with prepared programs at a rapid enough rate. This might be caused by several reasons. The card reader and a reasonable set of Tele-processing terminals may not be able to present programs to the IBM 7040 rapidly enough. The IBM 1301 disk file might create an untenable queue of requests for disk usage because of the time necessary for the access arm to seek new record locations. The IBM 7040 might be too slow to do the preliminary editing and preparatory

processing. The General Purpose Systems Simulator was used to answer precisely the question of keeping the IBM 7090 busy. This is one of the questions it is necessary to ask in order to select a feasible system. It is *not* a question of economic feasibility or evaluation relative to alternative systems. A different model might aid such evaluation, but the one described below would be of little use to the systems engineer for evaluation. Moreover, the present model does not serve well to answer the question of output congestion. It was chosen simply to illustrate that each simulation can accomplish only specific, narrowly defined objectives, and none can be all inclusive. Certainly, a more detailed model could aid in answering more questions, but it is often more practical to build a model to answer a few specific questions because of the simplicity this course affords in the model design as well as interpretation of the results of its manipulation.

model construction

Possibly the first step in designing the model we desire is to identify the transaction unit. It should be identified as a unit of information which is presented to the computer complex for processing. It could be a single character, an 80 character record, or a full program of variable size. A myriad of choices is possible. It is this choice that determines the level of detail of the model. In general, the smaller the transaction unit, the greater the number of entities of reality (and relationships between them) that have to be identified and represented in the model. Thus, model complexity is increased with the benefit (if the entities and relationships are correctly identified with reality) of a closer approximation to reality by the model. The color of the disk file frame is a property of reality we obviously do not want represented in our model. It is not obvious, however, that our model need not reflect the fact that seek time for the disk access unit depends upon the location of the read-write arm relative to its desired location. (For our purposes the use of an average time is sufficient.) The successful application of modeling depends upon the judgment of the systems engineer in choosing an appropriate degree of model detail, what to ignore, when to use averages, etc. Again, this judgment is greatly aided by clear specification of objectives in advance.

identification of transaction unit

The sub-program was chosen as the transaction unit. It is homogeneous in that it consists solely of one type of information (Fortran, FAP, binary program or data). It is necessary to maintain separation of types of sub-programs on input because of their different processing requirements. For example, a transaction representing a Fortran sub-program might require, on the average, 432 milliseconds to edit and condense preparatory to writing it onto the IBM 7090 input tape, while one representing a binary sub-program might require 1000 milliseconds to prepare it in a relocated absolute form.

assignment of parameters to transactions

Each transaction, therefore, was assigned a parameter value upon generation which indicates its program type. This parameter value was carried with the transaction and was appropriately used to guide the course of processing for that transaction. Further-

more, at times it was found necessary to tag transactions with parameters representing other attributes of sub-programs as well. Values for these parameters were also assigned upon generation of each transaction and carried with it until its termination. Some of these other parameters represented the following:

1 Number of sub-programs (transactions) which comprise the main program to which the transaction belongs.
2 Identification (job number) of the main program to which the transaction belongs.
3 Number of input cards represented by the transaction.
4 Number of disk tracks necessary to store the transaction on input.
5 Number of tape records necessary to admit the transaction on input.
6 Number of lines to be printed as a result of processing the transaction.
7 Number of cards to be punched as a result of processing the transaction.

competition for facilities

Facilities which were identified included the IBM 7040 central processing unit, a disk unit channel, a card reader buffer, a tape channel, a card punch buffer, two printer buffers, and the IBM 7090 central processing unit. Distinct processing operations compete for the use of these facilities. For example, at a given moment the disk channel could be called upon to help perform the following:

1 Store in the disk storage unit a request from a remote terminal to run a library program.
2 Store a portion of a Fortran sub-program to await accumulation of the complete sub-program being entered into the system from the card reader.
3 Store the results of a program already executed by the IBM 7090 in order to await the availability of an output device.
4 Present the IBM 7040 with the results of a previously processed program for punching.
5 Present the IBM 7040 with the results of two previously processed programs for printing by the two printers.
6 Present the IBM 7040 with a portion of a sub-program for editing and submission to the IBM 7090 input tape.

It is precisely the extent of this kind of interference and competition for facilities that we wish to gauge by simulation.

processing streams

When isolated from each other, processing operations of the above types are comparatively easy to analyze at least to the depth of detail necessary to specify their demands upon the identified facilities. A separate model was built for each of the following nine processing operations called "streams" because they represent a flow of transactions:

1 Card reader to disk storage.
2 Remote terminals to disk storage.
3 Disk storage to IBM 7090 input tape.

4 IBM 7090 processing.
5 IBM 7090 output tape to disk storage.
6 Disk storage to printer # 1.
7 Disk storage to printer # 2.
8 Disk storage to card punch.
9 Disk storage to remote terminals.

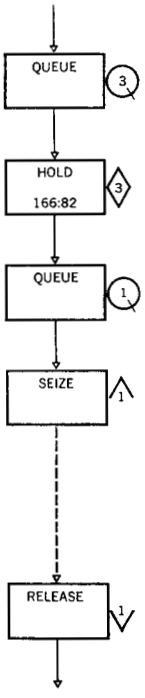
Some information flow (error messages, etc.) is not included in the above, but the density of this traffic is so slight that it was ignored.

The model for each of the above streams generated its own set of transactions either as fast as the system could accept them or at a rate determined by previous study to be about that which one would find in real life. For example, for the disk storage to input tape stream, a new transaction would be generated as soon as the previous one was completely written on tape; whereas for the card reader to disk storage stream a transaction was generated every minute (actually, whenever the simulator program clock counter reached a multiple of 60,000).

queues

The simulator program manipulated these independent stream models concurrently. That is, as the basic clock counter in the program stepped through discrete intervals of time, transactions were generated for each of the streams and logically transported through the blocks of the models at the rates each dictated. Contrary to indications above, the stream models were not actually independent since each referred to the same facilities. As a consequence, unless an interrupt was allowed, a facility in use by one stream could not be used by another until the first released it. Queues of requests for use of facilities formed of transactions from several stream models. Thus, the simulator program performed the service of linking the "independent" stream models and collecting statistics on the interference caused by this linking.

Figure 2 Typical block chain



In order to illustrate how this is accomplished, consider the chain of blocks in Figure 2. This or a similar chain of blocks may exist in any of the stream models. The numerals 1 and 3 to the right of the boxes are labels which reference the IBM 7040 central processing unit and the disk channel facilities respectively. Many QUEUE boxes at diverse locations in the total model may have the same label and, thus, refer to the same queue. A transaction entering the above chain will proceed directly to the second box (the HOLD box) if the disk channel facility is not being used (held) by some other transaction. If the disk channel facility is in use, the transaction will be held in the queue until it can proceed. Once it enters the HOLD box, a transaction remains there for an integral number of clock counts selected at random (by the simulator program) between 84 and 248. These clock counts correspond to the number of milliseconds the attention of the disk storage channel would be required in order to service that transaction. After waiting the appropriate time, the transaction would attempt to proceed through the other blocks of this chain. It may be held in queue # 1 if the IBM 7040 central processing unit

facility is in use, etc. The SEIZE and RELEASE pair of boxes perform the same kind of function as the HOLD box, but with the added feature of allowing the current transaction to traverse other boxes inserted between the SEIZE and RELEASE boxes, all the while holding in use the facility in question (the central processing unit in this case).

Interaction of the type illustrated clearly is effective between different parts of a single transaction stream as well as between separate streams. The ability to obtain interaction between seemingly independent portions of the total model as illustrated above is one of the keys to the power of the General Purpose Systems Simulator as a modeling tool for study of information processing systems. The simulator, therefore, incorporates *simultaneity of action* of different parts of the model with the concept of *sequencing through time* intervals.

possible model extension

As we have seen, the primary use of the model under discussion was to determine whether or not undue congestion would occur with each stream operating at its reasonably expected maximum rate. The construction of several stream models to interact when manipulated was simpler than the construction of a more realistic single unified model which would follow a program from entry to the computer complex completely through to the presentation of its processed results by an output printer, card punch or remote terminal. Such a "single stream" model could afford, in addition to more information about points of congestion, statistics on the total time a transaction (sub-program) remains in the system. Such statistics would be invaluable in the evaluative stage of the systems engineering process to assess possible benefits of this system in the area of reduced turnaround time and dynamic scheduling.

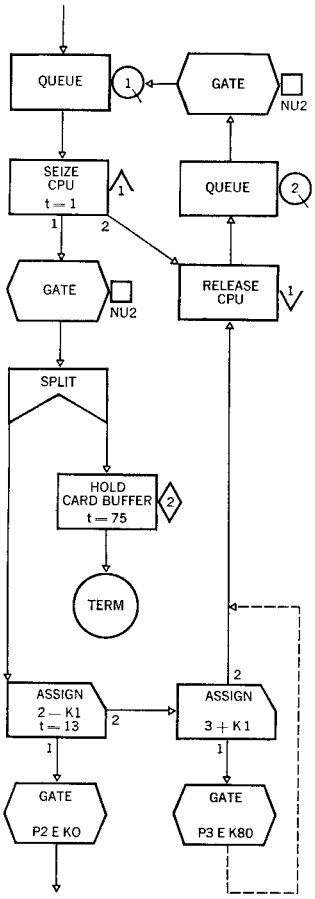
transaction decomposition

For some purposes it may be desired to break the transaction effectively into smaller units. This may be done either by splitting a transaction into two distinct transactions or by sending a transaction through a loop of the model a number of times equal to the number of subunits it represents. This may be illustrated in our example by a consideration of the timing of the reading of those cards which may comprise a sub-program entering the computer complex.

Assume that the value of the second parameter associated with a transaction (indicated by P2) represents the number of cards in the sub-program and has been assigned prior to the entry of the transaction to the QUEUE block at the top of Figure 3. This is the queue of transactions awaiting service by the central processing facility. When that facility is available, the transaction will move to the SEIZE block to obtain service from the facility for one unit of time. It will then attempt to enter the GATE block and proceed to the SPLIT block to be divided into two distinct transactions.[2] The gate allows passage only if facility number two (the card reader buffer) is not in use, as indicated by the flag NU2.

One of the two transactions leaving the SPLIT block holds the card reader buffer facility for 75 clock counts (milliseconds) and

Figure 3
Simulator diagram



then terminates. The other proceeds to the first ASSIGN block where it remains for 13 clock counts (an estimate of the time required of the central processing unit to accept the information read from one card). At this point the value of parameter two attached to the transaction is decremented by the constant one, (as indicated by "2—K1"). The transaction then attempts to pass through another gate which allows passage only if "P2EK0," that is, the value of the second parameter has been reduced sufficiently to equal the constant zero. This is the exit used when the transaction has traversed the loop a sufficient number of times to denote the reading of all of the cards represented by the transaction.

If the value of P2 is not zero, the transaction proceeds to the second ASSIGN block, wherein the value of parameter three is incremented by one. This parameter initially has the value of zero and is used as a counter. When this counter attains the value 80, the gate below this ASSIGN block allows the transaction to exit to a chain of blocks which represent the writing into disk storage one disk track record containing the information from 80 cards. (An odd number of cards is processed by a routine not shown in Figure 3.) It then returns to the loop to release the central processing unit facility and continue circulating. If the counter P3 does not have the value 80 then it immediately moves to release the central processing unit facility. From there the transaction attempts to return to the entry point of the loop, but may wait in a QUEUE block behind a gate until the card reader buffer facility has been released.

The direct path from the SEIZE block to the RELEASE block is taken only the first time the transaction circulates through the loop, and then only in the unlikely case the card reader buffer facility is still held by the previous transaction. It should be noted that by logic not shown in Figure 3, only one transaction is allowed to circulate in the loop at a time. Clearly, all the cards for a sub-program must be read before those of another sub-program can be entered into the complex.

simulator vs. flow diagram

The consequence of this procedure is that a transaction may assume at various times the roles of a single card or a record of 80 cards as well as its primary role of a complete sub-program. Note that when split, one of the two resultant transactions remained existent only long enough to hold the card reader buffer facility a specified number of clock counts. The apparent equivocacy of the meaning of the transaction reveals a fundamental aspect of the kind of models of information processing systems which can be built in the language of the General Purpose Systems Simulator. The block diagram describing the model is deceptively like but fundamentally different from a flow chart which might be drawn to portray the processing logic for programming purposes. In the real system under consideration, no information unit recirculates through a processing loop to effect card reading and no information unit is created for the purpose of timing the use of the card reader buffer. Indeed, even though we have called for the identification of the transaction with a

unit of information to be processed (the sub-program), we now see that this is an inaccurate but convenient device which serves only to help the systems engineer to understand better the animation of his model. *The transaction is simply an artifice to produce this animation.* It may be identified with different units of information, or none at all, at different points in its flow through a model.

simulation results

Partial results of the simulation of the scientific computer complex are presented in Tables 1 and 2 without the delusion that they might be applicable to any particular installation, but given only to illustrate the kind of numeric results available from the program. These are results from model manipulation corresponding to 104 minutes of time. The statistics for facility number 7, the IBM 7090 central processing unit, indicate that only five transactions held this facility, and they each held it approximately twenty minutes (1,200,000 clock counts). Here again the transaction did not correspond to a sub-program (in knowing violation of our earlier implication), but represented the amount of work presented to the IBM 7090 on a single magnetic tape. The IBM 7090 stream model did not purport to mirror "green light" time, etc., but served simply to time the transmission of magnetic tapes between the two processors. Consequently, the 100% facility utilization figure is from one point of view very unrealistic, but from the point of view from which the model was constructed, it is precisely what was desired and in that sense realistic. Caution in the interpretation of results is again signaled.

The usage of a facility is best judged in terms of its average utilization coupled with a history of the queue of requests for its usage. For example, Table 1 shows that the IBM 7040 central processing unit was in use 76% of the time available and that at least once there were seven requests for its use waiting. This is the maximum number possible according to the way the model was designed. The disk channel, on the other hand, was busy only

Table 1 Simulation results

Facility No.	Name	Average utilization	Number of entries	Average time/trans.
1	7040 CPU	0.7639	122,616	39.22
2	Card Reader	0.1224	10,272	75.00
3	Disk Channel	0.4835	18,363	165.75
7	7090 CPU	1.0000	5	1,259,081.39
8	First Printer	0.6547	41,219	100.00
9	Second Printer	0.6545	41,205	100.00
10	Card Punch	0.0714	1,872	240.00

Queue No.	Name	Maximum contents	Average contents	Total entries	Zero entries	Per cent zeros	Average time/trans.
1	7040 CPU	7	1.31	122,616	74,367	60.7	170.87
2	Card Reader	1	0.10	10,173	89	0.9	62.00
3	Disk Channel	4	0.19	18,363	11,013	60.0	161.81
8	First Printer	1	0.42	41,219	4,755	11.5	72.43
9	Second Printer	1	0.42	41,205	4,773	11.6	72.49
10	Card Punch	1	0.05	1,872	417	22.3	224.31

Table 2 Queue of transactions waiting service from disk channel

Upper limit	Observed frequency	Per cent of total	Cumulative percentage	Cumulative remainder	Multiple of mean	Deviation from mean
1	17	0.23	0.2	99.8	0.006	−1.407
51	1075	14.63	14.9	85.1	0.315	−0.969
101	1475	20.07	34.9	65.1	0.624	−0.532
151	1539	20.94	55.9	44.1	0.933	−0.095
201	1114	15.16	71.0	29.0	1.242	0.343
251	735	10.00	81.0	19.0	1.551	0.780
301	475	6.46	87.5	12.5	1.860	1.218
351	375	5.10	92.6	7.4	2.169	1.655
401	241	3.28	95.9	4.1	2.478	2.092
451	138	1.88	97.7	2.3	2.787	2.530
501	85	1.16	98.9	1.1	3.096	2.967
551	41	0.56	99.5	0.5	3.405	3.404
601	15	0.20	99.7	0.3	3.714	3.842
651	14	0.19	99.9	0.1	4.023	4.279
701	4	0.05	99.9	0.1	4.332	4.717
751	3	0.04	99.9	0.1	4.641	5.154
801	2	0.03	100.0	0	4.950	5.591
851	0	0.00	100.0	0	5.259	6.029
901	1	0.01	100.0	0	5.568	6.466
951	0	0.00	100.0	0	5.877	6.903
1001	1	0.01	100.0	0	6.186	7.341

Entries in table 7350 Mean argument 161.814 Standard deviation 114.318

48% of the available time and had at most four transactions waiting in a queue for its attention. Note, however, that although the central processing unit had a greater utilization and at least once had a longer queue waiting for its service, these facilities each granted immediate service to the same percentage of the number of requests for its attention. This is seen by noting that approximately 60% of the entries to these queues were "zero entries," that is, they did not wait at all in the queues. It should be noticed in passing that for a facility, the average time per transaction is computed as the total clock time that facility is in use divided by the number of transactions which entered it. For queues, the average time per transaction is the average period of residence in each queue computed only for those instances wherein the transactions remained in the queues a non-zero length of time.

Table 2 refers wholly to the queue of transactions awaiting service from the disk channel facility. An entry in the second column indicates the total number of transactions that had to wait in the queue an amount of time (measured in millisecond clock counts) between the time indicated as the limit time on the preceding line, and the upper limit time on that entry's line. This, then, is a tabulation according to frequency classes. We see, then, that 55.9% of the transactions entering this queue had to wait less than 151 clock counts for service, 95.9% waited less than 401 clock counts and, interestingly, one transaction waited almost a full second (more than 951 clock counts). The limits specifying the classes were chosen by the systems engineer as a part of his model design. The degree of detail is, thereby, according to his purposes and judgment.

Would this justify the acquisition of a second disk storage

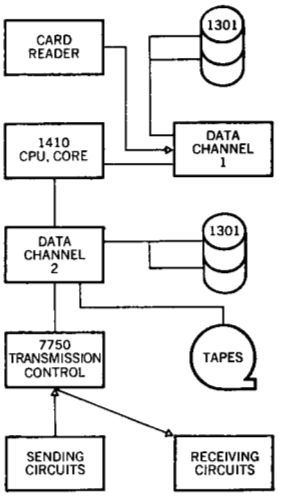
access arm? This can be answered only in terms of the human measurement values which should have been defined prior to the simulation. Full cognizance should be taken of the precise purpose of the simulation. As we have seen for this case, the simulation was undertaken only to see if congestion becomes serious enough to prevent the IBM 7040 from submitting enough work to the IBM 7090 to keep the latter busy. Since congestion is not this serious, and since we then do not care if, for the kind of work to which the system will be put, a unit of work waits for one or several seconds for disk service, the reply to our question is "no." Clearly, the same results displayed in Table 2 could generate a "yes" reply in a different milieu with, possibly, more stringent real-time Tele-processing requirements.

implications of the simulation

Second example : message handling

A second system example may serve to illustrate the use of the General Purpose Systems Simulator to examine an information processing system whose critical components are found to be external to IBM equipment. This system was designed to effect the message transmission functions between the more than 40 offices of a stock brokerage firm. It contains more than 10 fully duplexed communication circuits emanating from a centrally located IBM 7750 Programmed Transmission Control Unit. Several of the remote offices are served by each communication circuit and, consequently, compete for circuit service. The IBM 7750 is appended to an IBM 1410 Data Processing System with two modified IBM 1301 Disk Storage Units. The basic information flow is indicated in Figure 4.

Figure 4 Basic information flow



The capabilities of the communication circuits are the critical factors in the determination of the overall system capacity. The purpose of the simulation was to help evaluate different system topologies produced by variations in the allocation of offices to circuits. The simulation contribution to this evaluation was made in terms of the amount of time various types of messages required for transmission and processing and the amount of time they might be required to wait for service because of message congestion. More than thirty message types were identified as transactions for the simulation model. These message types included purchase orders and execution reports for stock listed on either the New York or American stock exchanges, commodity purchase orders and execution reports, over the counter purchase orders and reports and price requests and quotes. They also included inquiry and correction messages as well as other administrative information.

model construction

In contrast with our preceding system example for which only the most gross specification of terminal and data communications channel requirements was necessary, simulation of the brokerage system required that much attention be given to the communications circuits, the scanning discipline and the message assembly and disassembly performed by the IBM 7750. The set of entities of the real system identified as facilities in the model included:

1 The communications channels.
2 The IBM 7750 processing unit.
3 The data channel linking the IBM 1410, the IBM 7750 and one of the IBM 1301 disk storage units.
4 The IBM 1410 processing unit.
5 The data channel linking the IBM 1410 and the other IBM 1301 disk storage unit.
6 The two arms on each of the IBM 1301 disk storage units.

The magnetic core storage within both the IBM 1410 and the IBM 7750, as well as each module of the disk storage units, were identified as stores for the model. This model also consisted of several streams, each generating, moving and terminating its set of transactions. While the streams in the model of the scientific system represented, in general, different processing stages for the same data, each stream in the brokerage model served the entire life of the messages with which it was concerned. Consequently, a transaction representing a message could be tagged when generated with the clock time of its origination. This, when subtracted from the clock time at its termination, produced the time of its life within the total system which was tabulated. The streams of the brokerage model also differed from those of the scientific model in their linkage. In addition to sharing common queues, facilities and stores, they also shared many chains of blocks in the model representing processing steps common for all types of message transactions. Separation of the streams upon exit from such common chains was accomplished by usual logical branching techniques.

validity of input data

Results of a simulation procedure are never any better than the quality of the raw data upon which the model is built. We have already noted in the first example that a number of quantitative statements must be made regarding time, size of data units and frequency of occurrence. It was assumed that a new sub-program entered the card reader on an average of once every minute, that a Fortran sub-program required an average of 432 milliseconds to edit preparatory to writing on the IBM 7090 input tape, that a tape record would be written in approximately 54 milliseconds, etc. The assignment of program type to a newly generated sub-program was made on a random basis with a specified probabilistic distribution by means of a pseudo-random number generating routine within the General Purpose Systems Simulator program. The size of each sub-program was similarly determined. The data used was assumed to be typical and yet not representative of any particular installation. The brokerage study, however, was conducted for a specific firm. Many weeks were spent by systems engineers sitting beside Teletype operators logging messages, noting their types, lengths, points of origin, time of initiation, etc. Such historical data needed to describe present operations for comparative purposes and for statistical extrapolation to future requirements simply was not previously available in a usable form. This situation is probably more ex-

emplary of the rule rather than the exception and exemplifies another aspect of human interaction with the total systems engineering process.

Sometimes, even after an effort such as that for the brokerage study, the historical data available to a systems engineer for a simulation may be adjudged insufficient to allow him to describe the input to the system accurately enough to induce confidence in his results. Several courses of action remain open for him. The simulation program may be rerun several times, each time with different choices for transaction generation times, parameter assignment and their stochastic distributions. If this is not sufficient, he should consider developing a model and simulating the *present* system to the same approximate detail as the model of the proposed system. Then the results of that simulation with the questionable input descriptions can be compared with actual observed performance of the present system to partially validate the input descriptions. On other occasions a statistical description of input data is impractical, and the only reasonable procedure would be to furnish to the simulator as input a list of jobs presented to the present system, together with a description of the pertinent attributes of each. The General Purpose Systems Simulator does not permit this at the present time. For this reason some systems engineers have found it advisable to write a simulator program tailored to their specific system, or at least to those very similar to theirs. This was done, for example, to examine a proposed multiprocessor system for a firm in the aerospace industry when it was desired to obtain statistics on the predicted performance of the proposed system when processing a specific set of jobs. This set consisted of all (approximately 2000) of the jobs generated by the firm in a specified six day period. In this case it took several months to collect, check and prepare this information as input to the simulator.[3]

One of the first general discrete systems simulator programs was the Job Shop Simulator (M and A-1) written for the IBM 704 in 1959 by the Mathematics and Applications Department of the Data Systems Division of IBM. While that program is limited to apply only to those systems which are logically similar to a job processing machine shop, it did allow the independent preparation of a history tape to be used as the generator of transactions as well as their synthetic generation. Overall, however, the General Purpose Systems Simulator is indeed amenable to more general application than the Job Shop Simulator.

necessity of program description

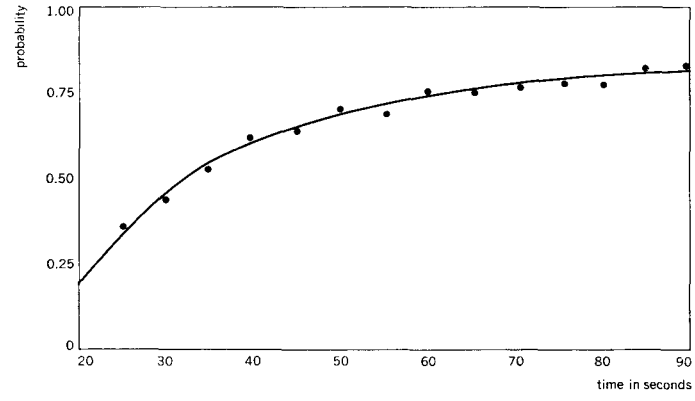
In addition to data concerning the information to be processed, accurate estimates of the number, sizes and speeds of operation of programs necessary to process each transaction type have to be supplied by programming specialists for the construction of a model of an information processing system. This included both IBM 1410 and IBM 7750 programs for the brokerage study. Furthermore, it was necessary to specify in considerable detail the functional characteristics of the supervisory control program, since some types of messages would require but one disk storage

access, while others would require four or five, etc. Such information closely relates to the question of congestion points within the system, and, consequently, to the total system capacity. In this case the specification of these operating characteristics was more difficult than the actual model construction. Determination of such operating characteristics for the scientific computer complex model was easier partly because of greater prior knowledge of required processing steps, and partly because the depth of detail of the simulation was not as great. That simulation did not provide information about the elapsed time a program was in the system; such information was the prime goal of the brokerage simulation. Detailed specification of the supervisory control program to reside in the IBM 7040 would be required in order to obtain such information for the scientific system.

results of the simulation

The result of the brokerage study was that the proposed system could easily handle the projected peak hour message traffic without serious delay of any messages. The predicted amount of time a message would spend in IBM equipment never exceeded .8 seconds and increased only slightly as the peak hour trade volume increased. The predicted transit time in the communications circuits, however, was up to 100 times as great and was very sensitive to increases in traffic volume. Since a tabulation was kept of the message life span for each message transaction, curves, such as that in Figure 5, were easily obtained to estimate the probability of the total transit time of a message being less than any given time.

Figure 5 Probability of transit time being less than a given time



Evaluation of a proposed information processing system invariably entails an estimation of the cost of standard equipment, special engineering, physical installation, personnel training, special systems programming, daily operation, etc. Cost is a very human subject, liable to wide variations of interpretation and evaluation. Hence, again we note the human values aspect of systems engineering. The brokerage system requires little special engineering, but may require retraining of many people. The scientific system requires no retraining of users, but requires ex-

tensive special engineering. Indeed, the effective monitoring of the IBM 7090 by the IBM 7040 depends upon some sort of direct communication between the two processors. Details of this communication are not germane to our purposes here; suffice it to say that examination of the special engineering required is extremely important within the larger systems engineering context.

The brokerage system requires a special supervisory program, but this will probably not change appreciably with time Systems programming is, therefore, primarily a one time cost. The scientific system, however, requires not only a supervisory monitor, but changes in other systems programs (such as the Fortran compiler) as well. Since these are evolving entities, the maintenance of these systems implies recurring costs. These are the changes that would effect greater utilization of the IBM 7090, but does the gain overcome the cost? Again, the question is not directly concerned with simulation, but is a part of the larger view.

relative cost

Absolute cost is meaningless; only relative cost has content. Systems proved feasible in a systems engineering study must be compared on a cost basis. Should the communication for mass data transfers between the processors of the scientific complex be by means of a random access disk unit instead of switchable magnetic tapes? Is the performance gain effected by having the remote terminals directly connected to the computer by means of the IBM 7750 worth the cost of this unit? Simulation may help measure performance; the systems engineer must measure cost.

Finally, the cost of a proposed system must be compared against the cost of no system at all, the cost to society of not performing the full desired function.

Classical mathematical analysis seeks to represent a discrete reality with a continuous model. The digital computer presents a new analytic vista. It permits analysis of discrete systems in a much more realistic manner than was permitted by earlier formal methods. Furthermore, it permits analysis of much more intricate systems. The systems engineer can now conduct elaborate statistical experiments within a computer. He can become an empiricist without the normally attendant cost of physical experimentation.

ACKNOWLEDGMENT

The scientific multiprocessor complex described in this paper has been developed in some detail by Mr. A. E. Speckhard. The simulation was performed by Miss B. Broome. Both Mr. Speckhard and Miss Broome are members of the IBM Data Processing Division's Market Development group and both are located in San Jose, Calif.

The brokerage system was simulated by Mr. L. R. Esau of the IBM Data Systems Division. Mr. Esau is engaged in Systems Planning at Poughkeepsie, N. Y.

REFERENCES AND FOOTNOTES

1. G. Gordon, "A General Purpose Systems Simulator," *IBM Systems Journal,* this issue, p. 18.
2. The simulator program used is a modified version of the original General Purpose Simulator and allows for the exit of more than one transaction from a block and the attachment of several parameters to a transaction.
3. F. R. Baldwin, W. B. Gibson, and C. B. Poland, "A Multiprocessing Approach to A Large Computer System," *IBM Systems Journal,* this issue, p. 64.

BIBLIOGRAPHY

L. Gainen, "A Simulation Model for Data Systems Analysis," *Proc. of the Eastern Joint Computer Conference,* Washington, D. C., December 1961, The Macmillan Co.

G. Gordon, "A General Purpose Systems Simulation Program," ibid.

C. Hammer, "Computers and Simulation," *Cybernetica,* v. 4, # 4, (1961).

Proceedings, Symposium on Digital Simulation Techniques for Predicting the Performance of Large Scale Systems, May 23 to 25, 1960, Ann Arbor, Michigan, Report # 2354-33X, University of Michigan.