1981 Winter Simulation Conference Proceedings
T.I. Ören, C.M. Delfosse, C.M. Shub (Eds.)

BARRIERS TO THE PRACTICAL USE OF SIMULATION ANALYSIS

Harry M. Markowitz
IBM Thomas J. Watson Research Center,
P.O. Box 218, Yorktown Heights, N.Y. 10598

ABSTRACT

Simulation techniques are used in only a small fraction of instances in which they seem applicable. This paper considers the reasons for such "non-uses." In particular the paper considers simulator programming, the simulation/database interface, and two statistical topics as past, present and future limiting factors in the practical use of simulation techniques.

1. INTRODUCTION

In the design and running of businesses--including manufacturing, marketing, finance and the like--simulation techniques are used in only a small fraction of instances in which they seem applicable in principle. True, many simulations are done per year; many simulation papers are given; conferences are held regularly. But it is the exception rather than the rule that businesses use simulation to answer "what if" questions; the exception rather than the rule that simulation is used to think through, for example, the consequences of changes in equipment or procedures. Simulation has rarely become an integral part of business in the way that linear programming has in petroleum refining, the standard actuarial model has in insurance, or double entry bookkeeping has universally in accounting for business receipts and expenditures.

There are various reasons that simulation is not used in particular instances. Reasons for such "non-uses" include the following:

(1) It is not practical (yet?) to simulate a particular situation for reasons that cannot be corrected by better training or software packages. Perhaps, for example, even with the most skillful aggregation and design of experiment, computer execution costs exceed possible benefits.

(2) Simulation would be practical, except that even the most skilful team would take longer to build the model, including data collection and analysis as well as programming and debugging, than the practical situation permits. If it is a situation which arises repeatedly, each instance requires more time to modify the model (again, including data collection and analysis as well as program modification and debugging) than is practical.

(3) Simulation is practical, the concepts and packages are available, but business is wise not to use simulation since skilled simulation analysts are not available.

(4) Simulation is practical; the concepts and packages are available; skilled analysts are also available; the only reason that simulation is not used is that management has not been "sold."

I will discuss limitations of the second type, where simulation is potentially practical but the time required to develop the model, including data collection and analysis as well as programming, is too slow for the problem at hand. Problems in area 3 and 4, including how to train fledgling simulation experts, and how to sell management when simulation is practical in fact, are no less important; but exploring problems of area 2 will be a sufficient task for today.

The opinions expressed here are not based an exhaustive survey, but on my own experiences and reflections. If these comments generate discussion and reflection which in turn effects concept and package development, then perhaps they will be worth the collective manhours, yours and mine, that have been or are about to be expended on them.

2. PAST

In the late 1950's and the beginning of the 1960's I believed that the problem was one of simulator programming. Indeed, it could take longer to program a simulation of a factory than it took to build the factory itself. I speak here of detailed simulations in which some portion of the simulated system is described fairly (or quite) literally. If you aggregated sufficiently you could, even then, simulate the whole world in less than the six days it reportedly required to build it.

The programming bottleneck was perceived by many and led to the simulation languages such as GPSS (Efron & Gordon 1964), CSL (Buxton & Laski 1962), GASP (Kiviat 1963) and SIMSCRIPT (Markowitz, et al. 1963) which appeared in the early 60's. I speak of the discrete rather than the continuous languages since the former are the ones usually applicable to what we refer to here as "business" as distinguished from "engineering" problems.

I will not attempt to describe any of these languages except to note, for use later, that the last three listed have an entity, attribute and set view of the simulated world. As of any instant in time the world is viewed as consisting of entities of various types. A given entity is described by the values of its attributes and the sets to which it belongs. SIMSCRIPT sets have owners as well as members; for example, in a jobshop simulator each machine group is said to own (have associated with it) a set of waiting jobs (usually called its queue); in a traffic simulator each stoplight could own a set of waiting cars. An entity, then, is characterized by the values of its attributes, the sets to which it belongs and the members of the sets it owns. An entity can have any number of attributes, own any number of sets and belong to any number of sets.

The simulation languages reduced manyfold the elapsed time and manhours required to program a simulator. In most instances the programming of the simulator is no longer the principal bottleneck in the development and use of detailed business simulation models. Granted, improvements in programming and especially in debugging are always welcome; granted that some large detailed simulations still require many weeks of programming; nevertheless, even in these large-scale cases, the principal bottleneck usually lies elsewhere.

In the 1950s and early 1960s, large, detailed simulation applications would be planned by a team consisting of us modeling and programming specialists plus specialists from the specific substantive area (e.g., manufacturing engineers). The modeling and programming specialists and the substantive experts decided together the contents of the model; the computer types went off and built the programs while the substantive experts arranged for the collection of the data. It took many months to program the model agreed upon, but it took approximately as much elapsed time and more manhours of various sorts to collect the data.

The advent of the simulation languages reduced programming manyfold, but left data collection the largest visible obstacle to the timely development of detailed simulation models. Part of the difficulty with data collection was that needed data was on pieces of paper rather than in computer sensible form. But even when data was on tape or DASD it was difficult to get to it. The substantive experts usually had to work through information systems programmers none of whom had a grasp of their system as a whole. They sometimes remembered the whereabouts of data whose use they programmed; but the uncovering of other data required a painful detective process.

The difference between the time it took for information system programmers to build or modify the database description of a system as compared to the time for a simulation modeler/program to build or modify a detailed simulation of the same system suggested the following theory: Allow business system builders to view their databases as representations of a world (the business and its environment, for example); allow them to describe this world in terms of entities, attributes and sets; and allow them to manipulate this representation with the same commands as are available to the simulation programmers. Perhaps then the programming of business systems would become as (relatively) simple as the programming of their simulation. Further, the structure of the system (specifically, its entity, attribute and set structure) would be as clear as a complex but well documented simulation; and therefore properly educated analysts with special needs, like the need to obtain data for a simulation analysis, would be able to pull out required data themselves much more easily than they could by working through a staff of programmers, none of whom understood the system as a whole.

3. PRESENT

Various business reasons delayed the testing of this theory. During the last three years, however, Ashok Malhotra, Donald Pazel and I at IBM Research built a database management system based on the Entity, Attribute and Set view. The name of the system is EAS-E, pronounced "easy" not "ease", and stands for entities, attributes, sets and events; see Malhotra et. al. (1980). The first large-scale application of EAS-E showed a reduction of source program of better than 5 to 1, and an even greater but difficult to quantify superiority in maintenance and evolution as compared to the prior system.

I will describe EAS-E briefly and then return to the topic of the simulation/database interface. But first let me make clear that EAS-E is not an IBM product or part of any IBM plan. It is an experimental language developed at IBM Research and applied only within IBM.

I will illustrate EAS-E in terms of its first application: a rewrite and extension of the Workload Information System of Thomas. J. Watson's Central Scientific Services (CSS). CSS consists of about 90 craftsmen who do glass work, electronics, etc., for Thomas. J. Watson's scientists and engineers. The old Workload Information System, written in PL/I and assembler, was difficult to modify or extend. In the first instance an EAS-E version was built to operate in parallel to the old version, reading the same weekly inputs and generating the same weekly outputs. The EAS-E based system duplicated the functions of the prior system with about one-fifth as much source code. It was then used to program functions--particularly tailored, interactive query and update functions--which would have been difficult to add to the prior system.

Exhibits 1,2 and 3 show programming used to print one of the CSS weekly reports. This looks like SIMSCRIPT II programming (Kiviat, et al. 1969), but remember this code exhibits attributes of database rather than simulated entities. The Exhibits should be almost self explanatory to those who know SIMSCRIPT II, but I will explain a bit for those who do not.

Exhibit 1:

```
ROUTINE TO START_NEW_PAGE_AND_PRINT_HEADING
START NEW PAGE
PRINT 7 LINES WITH PAGE.V, TITLE, SUBTITLE, DATE, AND WEEK THUS...
CSS Information System                                      Page ***
                   Central Scientific Services
     *****************************************************
*******************************              ********** **
CSS CSS  DEPT  CHARGE ENTRY COMPL ESTIMT PRCDNG TOTAL TIME CUSTOMER NAME
AREA JOB  NUM   TO   DAY  DAY     WEEK  TIME RMNING /AREA RESP.

RETURN   END
```

Exhibit 2:

```
PRINT 3 LINES WITH TASK_AREA_NUMBER, JOB_NUMBER, JOB_DEPT_NUMBER,
JOB_PROJ_NUMBER, TASK_ENTRY_DATE, TASK_COMPL_DATE, TASK_ESTIMATED_HOURS,
INHOUSE_HOURS_FOR_WEEK, TASK_INHOUSE_HOURS+TASK_VENDOR_HOURS,
TASK_ESTIMATED_HOURS - TASK_INHOUSE_HOURS - TASK_VENDOR_HOURS,
JOB_CUSTOMER, JOB_DESCRIPTION, TASK_EST_VENDOR_HOURS,
VENDOR_HOURS_FOR_WEEK, TASK_VENDOR_HOURS, TASK_ASSIGNEE_NAME, AND STAR THUS...
*** ***** ****  ****  ***  *** ***** ***** ***** ****** *************
*********************************** ***** ***** ******    ************ *
-----------------------------------------------------------------------------
```

Exhibit 3:

```
FOR EVERY GROUP IN CSS_GROUPS, FOR EVERY AREA IN GROUP_AREAS
          WITH AREA_TASKS NOT EMPTY AND AREA_TYPE ¬= VENDOR_TYPE, DO THIS...
   LET SUBTITLE = AREA_NAME
   CALL START_NEW_PAGE_AND_PRINT_HEADING
   FOR EACH TASK IN AREA_TASKS WITH TASK_COMPL_DATE=0, CALL PRINT_TASK_LINES
REPEAT
```

Exhibit 4:

```
FIND THE JOB IN CSS_JOBS WITH JOB_NUMBER = PROPOSED_JOB_NUMBER
  IF ONE IS FOUND...
     CALL REJECT (|JOB ALREADY EXISTS WITH SPECIFIED JOB NUMBER.|)
     RETURN
  ELSE...
```

Exhibit 1 is a routine which starts a new page and prints a heading on it. This routine is called in the printing of several different reports. The START NEW PAGE statement is self-explanatory. The seven lines (including the blank seventh line) following the PRINT 7 LINES... statement are printed during run-time just as they appear in the source program, except that the first variable, PAGE.V (automatically maintained by EAS-E) is printed in place of the first grouping of ***s, the second variable TITLE is printed in place of the second groupings of *s, etc. The PRINT statement in exhibit 2, slightly simplified from its CSS version, prints 3 lines containing the specified variables and expressions in the places indicated in the three form lines (last 3 lines of the exhibit). These lines will print data under the headings of exhibit 1. Most of the data being printed are attributes of database entities of type JOB and TASK.

I mention as background to exhibit 3 that tasks to be performed by CSS are divided into areas. These areas, in turn, are divided into groups. Thus the "Device and Mask Generation" group includes the "Electrochemistry" and "Mask Gen Lab" areas.

The first phrase of the exhibit (FOR EVERY GROUP IN CSS_GROUPS) instructs the compiled EAS-E program to have the reference variable GROUP point in turn to each member of the set called CSS_GROUPS. For each such GROUP, the next two phrases ("FOR... WITH...") instruct the executing program to look at each area in the group's areas that meet certain conditions. For each GROUP and AREA thus generated, the program sets the variable SUBTITLE used in the START_NEW_PAGE_PRINT_HEADING routine, and then calls the latter routine. Next, under the control of a FOR and a WITH phrase, it calls on a subroutine which includes the PRINT 3 LINES... statement of exhibit 2 for tasks in the AREA_TASKS set which have not yet been completed. These six lines thus select groups, areas and tasks for which heading lines and body lines are printed for one of the CSS weekly reports.

The example of a FIND statement reproduced in exhibit 4, including the IF ONE IS (NOT) FOUND statement that usually follows a FIND, finds the job in the set of CSS_JOBS with its job number as specified. EAS-E has a different syntax for the FIND statement than does SIMSCRIPT II, to improve readability. Again, it is a job represented in the database, rather than a simulated job, which is found here.

In addition to the commands illustrated in the exhibits, EAS-E has SIMSCRIPT II-like commands to CREATE (the database representation of) an entity, FILE an entity into a set, REMOVE an entity from a set, do something FOR EACH in a set, define database entity types by telling EAS-E what attributes, ownerships and memberships EVERY entity of the type has, etc.

EAS-E allows the user to manipulate database entities much as SIMSCRIPT II allows him to manipulate simulated entities. But the mechanisms behind the scenes are necessarily different and often more complex. For example:

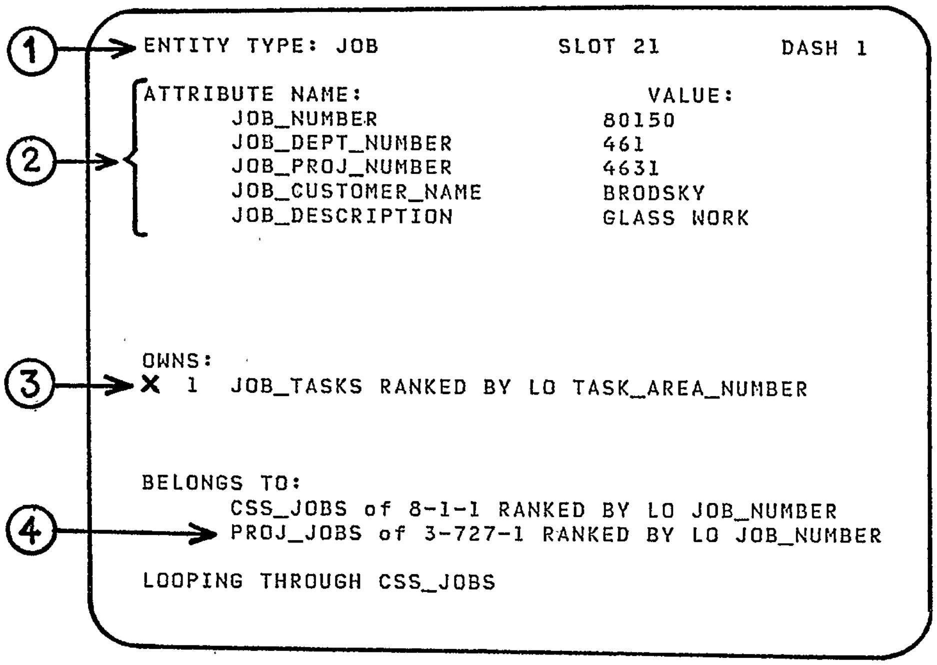
> (1) One or more users can interrogate or update parts of the database simultaneously. But an individual user's program does not read from or write onto the database directly, since one program might thereby clobber the work being done by another. Rather each communicates with a "Custodian" program which remembers which entities are already being accessed read-only or read-write.
>
> (2) The custodian must assure that if the system crashes at any point in time, either all of the changes or none of the changes of a given program are, in effect, reflected in the database. The programmer can ask to RECORD ALL DATABASE ENTITIES modified thus far; but again the custodian must assure that either all or none of these changes are recorded in case of a crash.
>
> (3) LIFO and FIFO sets are stored as linked lists as in SIMSCRIPT. But it would be very inefficient for large ranked database sets to be stored this way. For example, if a set of cars owned by a department of motor vehicles has a million members ranked by owners name, then on the average one-half million members would have to be accessed to FILE or FIND a car in the set if stored as a linked list. EAS-E actually stores the set by means of a balanced tree of subsets, in such a way that very few accesses are required to FILE or FIND in sets with millions of members.

The above goes on behind the scenes. The user still says CREATE, DESTROY, FILE, REMOVE, FIND, FOR EACH...WITH..., and the like.

Another difference between simulated systems and real business systems is that the simulated system whizzes along through time as fast as the computer can move it. The real system moves at a relatively more leisurely pace. Hence the analyst may wish to browse the current status of parts of the database. Towards this end EAS-E provides the analyst with direct (nonprocedural) abilities to browse the system and, if authorized, to update it directly. Similar facilities might be of use in simulated systems, perhaps to assist debugging, if combined with the ability to stop the flow of simulated time at designated events, times or conditions.

Exhibit 5 is an example of a Browser display. In this case the attributes, ownerships and memberships of one entity is shown on the analyst's screen. The first line of the screen shows the type of the entity plus certain internal identification information which we will ignore here. The region marked (2) shows attributes of the entity. We see, for example, that the JOB in the exhibit has JOB_NUMBER = 80150, JOB_DEPT_NUMBER = 461, etc. The line marked (3) tells us that the JOB owns a set called JOB_TASKS, that the set for this particular job has only one member, and that the set is ordered by job number. The line marked (4) indicates that the job belongs to a set called PROJ_JOBS.

You move through the database with the aid of the PF (program function) keys whose template is shown in exhibit 6. To look at every task in the set JOB_TASKS, for example, place an X as shown at (3), then press PF10 labeled FORWARD on the template. This brings the first (and in this case, only) task of the set. Generally, to move forward to the first or next member of a set press PF10, to move backwards in a set press PF11. Further capabilities include scrolling up or down



```
ENTITY TYPE: JOB                    SLOT 21        DASH 1
ATTRIBUTE NAME:                        VALUE:
      JOB_NUMBER                    80150
      JOB_DEPT_NUMBER               461
      JOB_PROJ_NUMBER               4631
      JOB_CUSTOMER_NAME             BRODSKY
      JOB_DESCRIPTION               GLASS WORK

OWNS:
X  1  JOB_TASKS RANKED BY LO TASK_AREA_NUMBER

BELONGS TO:
      CSS_JOBS of 8-1-1 RANKED BY LO JOB_NUMBER
      PROJ_JOBS of 3-727-1 RANKED BY LO JOB_NUMBER

LOOPING THROUGH CSS_JOBS
```

Exhibit 5

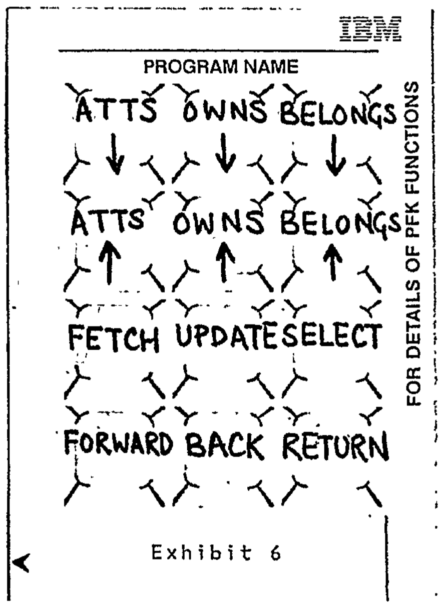


Exhibit 6

the displayed attributes, ownerships or memberships of the entity if there are more than can fit in the space allotted on the screen, specifying selection conditions to be used in looping through a set, and modifying attributes by overwriting their values on the screen.

My colleagues and I believe that the experience of EAS-E thus far will prove typical, and that application development can be reduced manyfold by a language (perhaps EAS-E, perhaps something like it) which uses the same worldview and basic concepts as some simulation language, and allows the application developers to manipulate database entities in the same way that simulation programmers manipulate information about simulated entities. This should help the simulation/database interface in two ways. First, it should be easier for any analyst with special needs to learn as much of the structure of the database as he needs to know, to browse the database and to program routines to pick up desired data. Second, the simulation programmer in particular should be helped by having database facilities which use the same general worldview, some of the same entity, attribute and set structure and perhaps even some of the event coding as will appear in his simulator.

4. FUTURE

Assume for the sake of the current argument that the problem of getting to the data stored in databases is "solved in principle." Imagine that such software and the knowledge to use it were widespread. What then would be the principal bottleneck to the application of simulation techniques? I believe this bottleneck would be of a statistical nature.

For example, rather than using past demands for products as exogenous inputs to a model, an analyst may want to generate demands randomly. This allows the analyst to test the effects of shifts of demand by changing parameters of the distribution, but it raises questions as to the form and parameters of the demand distribution. In particular, what form of model and values of parameters would explain past demands reasonably well?

I will not attempt to review the large body of good and relevant work on statistical analysis related to simulation, and beyond that the ocean of literature relevant to statistical inference in general, and hence to simulation in particular. Here I would like to point out what seems to me two major bottlenecks limiting the application of such techniques in practical business simulation analysis.

The first limiting factor has to do with packaging; the second is a philosophical matter with far ranging implications.

The notion of packaging may be illustrated by random number generators. As long as a random number generator is buried in the literature it is frequently of little use to the real-world analyst who must put models together quickly. In contrast, when a random number generator can be invoked by writing RANDOM.F or NORMAL.F or GAMMA.F, etc., then it is conveniently at the disposal of any simulation analyst with the background to understand the distribution generated.

The same applies to the analyst above who needs to select a model of demand and estimate its parameters. Ideally he should be able to specify the model and the data to be used in the analysis in a manner most convenient to himself, and have the rest done for him. Too often the world offers him much less. I realize that I am only offering a gripe rather than a solution. But it is a complaint which I believe many in the field will confirm as identifying a limiting factor. Furthermore, this limiting factor could be greatly alleviated in this age in which compilers can deal with just about any unambiguous interface one cares to devise.

The next problem area concerns a basic philosophical matter. The business application of simulation analysis usually involves choice under uncertainty. No one tells the analyst the form of the demand distributions, much less their parameters. The object of the simulation analysis is to help guide action, even in circumstances when objective statistical analysis cannot conclusively reject all but one plausible hypothesis.

Without trying to untangle how much was already understood by Bayes, Ramsey, Jeffreys, and De Finetti, as cited in Savage (1954), one may say that certainly the most influential, and probably the most important work on action under uncertainty was that of Leonard J. Savage (1954). On the whole, however, the literature on statistical methodology as applied to simulation analysis is surprisingly oblivious to Savage's revolutionary ideas.

Savage provided a convincing, axiomatic defense of a position which includes Bayesian inferences as a consequence. The conclusion which Savage draws from his axioms is that, if a decision maker is to avoid ridiculous behavior (contrary to the axioms), he must act as if he attaches probabilities to possible states of nature, and acts so as to maximize expected utility given these probability beliefs. As information accumulates he alters his probability beliefs according to Bayes' rule. If evidence is strong, in that the power of the test is great in the Neyman-Pearson sense, then persons of widely differing a priori beliefs will be drawn close together in their a posteriori beliefs. If the power of the test is weak, specifically if several hypotheses could well have generated the data, then beliefs among these will shift little.

If you have not studied the axiom system you cannot really make up your mind whether you should accept this "subjective probability" approach. I will not attempt to present the Savage axioms here. An example will illustrate one problem with using so called "objective" statistics (of the "old fashioned" Neyman-Pearson variety) and perhaps encourage the reader to consider seriously the Savage alternative.

Returning to our analyst who needs to develop a model of demand, perhaps the first model that suggests itself is that the demands for a particular product has, over the last k years, been drawn repeatedly from some (unchanging) normal distribution. Suppose he tests this hypothesis and accepts it; then builds a simulation model that assumes normality, and uses this model to determine policy. The analyst and his Company may be in BIG trouble.

The source of this trouble is twofold. First, recall that to accept a hypothesis on the basis of an objective test means merely to fail to reject it. If asked, the same data might also accept (i.e., not reject) all sorts of other plausible hypotheses. Perhaps if the analyst had assumed that the (unchanging) distribution was from the broad family of Pearson distributions, which includes the normal as a special case, and had used a Bayesian calculation given some a priori beliefs, the data might have swung belief away from the normal rather than towards it--even though the evidence was not strong enough for a Neyman-Pearson test to reject the hypothesis at the specified level of significance.

Once the analyst has "accepted" a hypothesis by an objective test the second step toward big trouble is to act as if the hypothesis, e.g. normality, has been proved with certainty. This is done by building it into a simulation model, then using the model to answer policy questions. For example, suppose that the estimate of the mean you give the model is m, and the estimate of the standard deviation is s. Suppose that, in effect, the model is offered a 10,000 to 1 bet that the observed demand will not exceed m + 10s. Since this has virtually zero probability for a normal distribution, the model will presumably accept the bet. But with other distributions, perhaps other distributions which would be accepted by the same data which accepted the normal, a 10s deviation from the mean is far from "virtually zero" (e.g., the Tchebychev inequality assures only that the probability does not exceed .01).

What the subjective probability approach decrees in the above situation is to shift probability beliefs, based on the data, according to Bayes' rule; and evaluate alternative strategies as if nature first drew a model randomly, with probabilities equal to the updated beliefs, then drew a history randomly from this model. In particular it would not assume that a 10s move was virtually impossible unless (roughly speaking) the analyst's a posteriori beliefs virtually ruled out all models that give such an event any perceptible probability.

Suppose that you could be sure (perhaps by construction) that an unchanging distribution had mean zero and unit variance, but all you knew about the form of the distribution was that the normal hypothesis had been "accepted", i.e., not rejected. Would you bet $10,000.00 of your own money to win $1.00. Of course not. This shows that your intuition is more Bayesian than are the statistical procedures commonly used in building a simulator. I believe that if you study the Savage axioms many of you will decide that your instincts are right and the procedures are wrong.

Simulation is sometimes used in situations where objective statistical tests are appropriate, e.g., when trying to determine facts about a given complex random process which cannot be solved analytically. Thus I cannot fault any particular methodological study because it is objective rather than subjective. But taken as a whole the literature on statistics for simulation is quite deficient in Bayesian analysis. Certainly the "well packaged" statistical facilities called for above should include Bayesian as well as "objective" capabilities.

5. POSTSCRIPT AND SUMMARY

Imagine a business in the 21st century when planning is more rational than it is today. In particular, top management no longer expects someone to hand it a point estimate of what the future will be, as the start of a process which produces a great plan if the predicted occurs, but can be disastrous if a plausible alternative occurs instead.

This 21st century planning process seeks a strategy which, in effect, is robust against adversity as well as profitable when the likely occurs. When the answer is not obvious and optimization techniques not applicable, simulation will be used to trace out the consequences for business performance of major contingencies such as shifts in demand, in price or in technology. Since a particular business in the year 20XY will usually be much like it was in 20X(Y-1), model modification will consume more resources than new model construction.

For large complex businesses, models of components of the business will be used for more detailed questions, and to test more aggregate versions to be incorporated into broader scope models. The internal database of the business, and available relevant external databases will be used to estimate relationships for models, and to obtain initial conditions and exogenous events for certain runs. In the case of some detailed simulation models, the same coding will run the simulated shop (or other business component) as runs the real shop.

I have not said whether the above business exists at the beginning of the 21st century, 19 years from now, or at the end of the century, 119 years from now. Above I have outlined my views as to the concepts and packages needed to reach this stage. I would be most interested in hearing your views on these matters.

REFERENCES

Buxton, J.N., & Laski, J.G., Control and Simulation Language, *The Computer Journal,* Vol 5, 1962, pp. 194-199

Efron, R. and Gordon G., A general purpose systems simulator program and examples of its application: Part I - Description of the simulator *IBM Systems Journal* Vol. 3, No. 1, pp. 21-34, (1964).

Kiviat, P.J., *GASP -- A General Activity Simulation Program,* Applied Research Laboratory, U.S.Steel Corp, Monroeville, PA, July 1963.

Kiviat, P.J., Villanueva, R., & Markowitz, H.M., *The SIMSCRIPT II Programming Language,* Prentice Hall; Englewood Cliffs, NJ, 1969.

Malhotra, A., Markowitz, H.M., & Pazel, D.P., *EAS-E: An Integrated Approach to Application Development,* RC 8457, IBM T. J. Watson Research Center, Yorktown Hts., NY 10598, August 29 1980.

Markowitz, H.M., Hausner, B., & Karr, H.W., *SIMSCRIPT: A Simulation Programming Language,* The RAND Corporation RM-3310-PR 1962. Prentice-Hall NJ, 1963.

Savage, Leonard J. *The Foundations of Statistics,* John Wiley and Sons, New York, 1954.