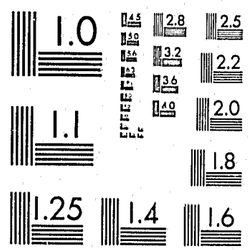

AN APPRAISAL OF PERFORMANCE MEASUREMENT IN SIMULATION SOFTWARE

by

Stuart Jay Deutsch

and

Jerry E. Richards

September, 1980

This work was performed under Grant #78-NI-AX-0003 from the National Institute of Justice. Points of view or opinions stated herein are those of the authors and do not necessarily represent the official position or policies of the United States Department of Justice.

ABSTRACT

Simulation software is shown to be generally deficient in its ability to measure the performance of system's simulation models. These deficiencies are described and it is determined that a unified measurement philosophy is what has been lacking in their design. Several symptoms of these deficiencies are suggested. The specialization of simulation software to particular applications, or system models, like Criminal Justice and marketing is in part symptomatic of the need for such a philosophy. Another reason is the requirement that simulation teams must often append measurement libraries to existing simulation software in order to augment the neglected measurement approaches available.

Having examined these deficiencies, we wish to see how a unified measurement philosophy might be used to overcome them. We define such a philosophy as a mix of performance measures, measurement strategies and measurement processes tailored to particular system behaviors, measurement goals and utilities for information. These characteristics are then developed into an approach for resolving the measurement inadequacies of simulation software based on the types of measures, measurement strategies and measurement processes which arise in the course of simulation projects.

A measurement model to be used for either simulation languages or simulators is proposed, and it is distinguished from related functions required of simulation software--e.g., report generation, measurement analysis, and system's model and experimental design formulation and execution.

CONTENTS

I. INTRODUCTION

The recent proliferation of simulation software, in the forms of new languages, simulators (loosely, general simulation models) and of input data processors and output analyzers, indicates the ever-present research opportunities directed at enhancing the practice of simulation. One area which has received little attention, except with regard to particular simulation models, is that of performance measurement. In contrast to output analysis in simulation, performance measurement concerns itself with the formulation and estimation of appropriate statistical models, called performance measures, from data judiciously collected during the running of a model. Output analysis, on the other hand, is generally concerned with the statistical analysis of performance measures already computed. Although both more-or-less require the creativity and modeling skills that are needed for the construction of the system's model, the measurement function should logically fall between the system's modeling, and the output analysis and documentation routines in simulation software. The existence of a major component in either general purpose simulation languages or simulators that does such work is, however, not to be taken for granted. They generally do not exist.

Even though one important reason for a practitioner's selecting the simulation paradigm is its considerable flexibility in modeling systems, the practice of the simulation art is often encumbered by high software development and usage costs. This is particularly true when new software is developed or old software is adapted to new application areas, but one approach taken by researchers to reduce

these costs has been to develop specialized simulation software for modeling new systems applications--like production simulators (e.g., Phillips et al., 1973), marketing simulators (Kotler, 1971, gives several examples) or criminal justice simulators (Chaiken, et al., 1975, and Deutsch and Richards, 1978, cite many such examples). This specialization is often unnecessary, however, since the mechanisms needed to model the system already exist in general purpose simulation languages and simulators (see Shannon, 1975, and Fishman, 1973). What prevents the general purpose software from being used for such systems partially depends on their lacking comprehensive performance measurement abilities. That is to say that the lack of a general mechanism in simulation software for constructing and analyzing performance measures limits their use for the variety of systems encountered in simulation projects.

Such deficiencies contribute to the misconception that existing software cannot adequately be used to describe some systems. In fact they usually do provide adequate system's modeling features, but the measures of performance which can be estimated do not facilitate the measurement of some marketing, criminal justice or many other systems possessing well-established (though, perhaps, not entirely adequate) philosophies of measurement.

This inability of simulation software to accomodate diverse measurement philosophies is just one factor that contributes to the development of software specialized to particular applications; nevertheless, the budgets of most simulation projects indicate that simulation software specialists should address these and other measurement problems which we shall indicate in the course of our discussions. The reduction of measurement costs and re-modeling costs for better measurement that accrue to practitioners should be one

goal of designers. What is needed to make these changes, however, is a philosophy of performance measurement that can be translated into effective measurement models, to be implanted in existing general purpose simulation software; that these models must be sufficiently flexible to accomodate a much wider spectrum of applications and of model behaviors, measurement goals and utilities for information than at present will be emphasized throughout our presentation.

Deutsch (1976) has already begun to develop a theory of performance measurement applied to criminal justice research, but our purpose here is to borrow from that theory in order to develop an appreciation for the effect that it could have on simulation practice through measurement-related changes in simulation software. To this end, we review the major elements of performance measurement theory and the limitations that current thinking on the measurement of performance of simulation models imposes on simulation practice. In our discussions of the existing problems with performance measurement approaches in simulation software, we consider the historical specialization of the measurement elements to the characteristics of quite limited conceptual models of simulated behaviors to be one of the most onerous to the practitioner. Because this emphasis has so severely narrowed the scope of the measurement approaches programmed in existing software, we propose a more flexible but rigorous simulation measurement theory to enable software developers to increase the useability of their offerings by relying on the tenets of performance measurement theory.

This paper begins in Section II with a brief review of the tenets of performance measurement theory. By describing the elements of a measurement approach--the measurement strategy, the measurement

process and the measures--we are later able to critique particular deficiencies of simulation software. However, our immediate goal for Section III is to describe an important symptom of inadequate measurement approaches in existing simulation software: The proliferation of new simulation packages specialized to either a particular project or at least a particular system.

Next, we explore some of the motives behind the kinds of measurement approaches found in existing packages. In Section IV, the measurement approaches of simulation software are seen as emphasizing two quite separate aspects of general systems. One emphasis is on one model of a particular system and a subset of its behaviors. The antithetical measurement approaches emphasize the behaviors of more general simulated systems, the goals of more general simulation measurement projects, and the utilities for information found universally in simulation measurement projects. In Section V, a survey of the measurement philosophies of existing simulation software shows that the latter orientation of measurement approaches has not been carefully considered. The evidence suggests that the performance measurement approaches that exist have been developed with a very limited interpretation of the measurement strategies, processes and measures which are appropriate for simulation projects. To make simulation packages more adaptable to the needs of particular measurement approaches and applications without having to program additional measurement features for each project, a conceptual model for the implementation in simulation software of the more general measurement orientation is offered in Section VI.

II. PERFORMANCE MEASUREMENT THEORY

The notion of performance measurement is one which is embodied in all empirical sciences. Deutsch (1976) presented a universal view of the ingredients necessary for the successful conduct of measurement studies. The theory of performance measurement, which he postulated for the development of a uniform measurement philosophy for the Criminal Justice System, was intended to supplant the ad hoc methods so often used in the design of criminal justice information systems and in the conduct of empirical research on crime. Since its recognition as a problem of Criminal Justice empirical research, a literature of performance measurement theory and application has been compiled (see Deutsch and Richards, 1979b), and efforts to improve the measurement of the performance of simulated Criminal Justice Systems have been expended (see Deutsch and Richards, 1979a, 1979c, 1979d, 1980a, 1980b, and 1980c).

As performance measurement is an applied science, the theory of performance measurement is actually a paradigm for the conduct of empirical science. The elements of performance measurement theory are the measurement processes, the measures and the strategies --all of which compose the measurement approach for particular applications. A measurement strategy is a policy that defines which data is to be gathered, when it will be gathered, and how much will be gathered. The measurement strategy may require the formulation of a model and the development and analysis of appropriate measures of performance based on the model's output, or it may require the measurements to be made directly on the system's output. The product of the measurement strategy is the measurement process whose output is the vector of performance measures.

The application of performance measurement theory has been described as requiring the development of a measurement approach consisting of:

1. performance measures appropriate to the behavior to be studied;
2. measurement strategies appropriate to the performance measures being applied and to the utility of the information supplied through measurement; and
3. specific structures or types of measurement processes that best support the chosen performance measures and measurement strategies, while meeting the purposes of the evaluation.

The practitioner must therefore provide the direction for a performance measurement project by first developing an understanding of the behavior of the system under investigation, of the purposes for his study, and of the utilities for the information which measurement may obtain. The forms of the measurement processes, measures and strategies thus evolve from the analyst's understanding of the system to be modeled and of his project and its goals (see Figure 1).

The application of performance measurement theory contrasts with the practice of less structured empirical research paradigms. In the more extreme, least organized examples of empirical research, empiricists often possess vague goals for the measurement project, and little or incorrect knowledge about the system's behavior; consequently, an appreciation for the true utility of information is not developed beforehand and the data is collected before the proper measurement approach has been designed. Such premature measurement is--to be sure--inefficient in addition to tending to

yield meaningless or (worse) incorrect information about the system. It is these kinds of deficiencies in empirical science for which performance measurement theory is meant to provide a structure, and hence efficiency and reliability to their research. The purpose of our examination of this issue vis-a-vis simulation software is to offer suggestions for providing such a structure for performance measurement which can carry with it similar advantages of increased efficiency and reliability for the conduct of simulation experiements and, consequently, for the reliability of the resultant conclusions.

III. SIMULATION SOFTWARE PROLIFERATION

Because simulation is an empirical tool used for the investigation of models and the behaviors of models, the practice of simulation frequently requires the project team to conduct experiments for measuring the effects of controlled changes imposed upon a computer model. The process by which these experiments are conducted--we shall refer to it as a measurement approach characterized by strategies, processes and measures--is fraught with the same difficulties that bedevil the practice of other empirical research methodologies. The practice of simulation, moreover, is littered with the remains of poor examples of performance measurement just like these other paradigms. Several reasons relating the failures of the measurement processes, measures or strategies to the failures of Criminal Justice empirical research were indicated by Deutsch (1976), but the blame for inconclusive simulation projects can all too often be attributed to the inadequacies of the software used. Annino and Russell (1979), for example, have provided ten reasons for project failures of which seven are either directly related to the software or could be ameliorated by better software features.

Although choosing the wrong computer software for a project when one ideally suited exists represents the problems of information, access or judgment faced by all research personnel, identifying software appropriate for simulation can be a problem because of the vast number of options available. If we consider that as early as 1972 Sammet (1972) identified 170 high level programming languages (software packages characterized by sets of instruction), any of which could be used for simulation projects, then we begin to see the difficulties--and the opportunities--presented by this selection.

Among those languages Sammet identified, several are specifically designed for simulation. Simulation languages like GPSS, SIMSCRIPT, SIMULA and DYNAMO are written in one or more of the so-called scientific programming languages like FORTRAN for modeling and simulating quite general systems models. Simulation languages have proved to be useful. Accordingly, their usefulness stems from the following reasons used to explain why a simulation language would be chosen over a general purpose programming language:

a. they require less programming time,

b. they provide superior error checking,

c. they provide a conceptional model for the system's model,

d. they provide software required to simulate the system's model,

e. they automatically generate certain required data,

f. they facilitate the collection and display of the data, and

g. they control the management and allocation of computer storage (Shannon, 1975; p. 106).

But the problem of picking the best software for a simulation project is not necessarily overcome by the choice of a simulation language over a programming language, because simulation languages are

themselves not always best suited for use in the conduct of some simulation projects. In such cases, either new simulation languages are developed to meet a project team's needs, an existing language is chosen and modified to their specifications, or the project may be abandoned entirely. Today, partially because of the failures of existing simulation languages, we have an abundance of software designed especially for computer simulation; however, some of the reasons for the phenomenal growth in the number of simulation programs that are available are generally the same as those offered by Sammet (1972) to explain "the incredible proliferation" of high level programming languages (see Table 1).

As Sammet points out about programming languages, true software innovations are those motivated by reasons (1) and (2) in Table 1, but they are not as common as the new programming languages motivated by the remaining five reasons. The latter software usually results from a programmer's seeking to combine known innovations into a new software package, or to either extend or modify the features of an existing package while developing an otherwise uninspiring software product. However, it is certainly true that innovations in simulation software also occur less frequently than do the other types of software developments (see Fishman, 1973, and Shannon, 1975).

Recently, for example, the development of simulation software has been directed at cultivating special applications areas. Often wanting both in the scope of the system's model which they can accomodate and in the sophistication of the software's ability to execute a model, these simulators are often little more than specialized

computer simulation models. Requiring only data to define a model's features and to execute a particular example, they neither provide nor do they require a set of language-based instructions that would be required to describe and execute a model with a simulation language. Simulators have thus become popular because they are generally easier to use than simulation languages. They also provide specialized abilities for simulating the classes of systems which can be modeled.

Nevertheless, a simulator is easier to use because the instructions of the language have been reduced to data, but this feature also restricts the flexibility and the variety of the models constructed for a simulator. Simulators suffer from this lack of breadth because more of the characteristics of the conceptual model that outlines a simulator's modeling abilities and limitations must be permanently encoded in the software to reduce the input from an instruction set to data. Thus, the more limited scope of the simulator's conceptual model relieves the vast amount of programming required to make it as flexible as a simulation language.

Ordinarily the switch to simulators from simulation languages would qualify as an innovation motivated by either reason (1) or (2) in Table 1, but the abundance of new simulators cannot be explained in this way. It seems that simulators are being developed--not out of a recognition of the need for, or the cost savings that result from, the use of such specialized software--but to a greater extent as a response to the inadequacies found to exist in simulation languages and in the more flexible simulators. In fact, the growth in the number of special applications software can quite often be attributed to the specialization of existing software attributes to the measurement needs of the

community of specialists that deal with the one application area. What occurs in the development of simulation languages and simulators today is that the measurement approach--that is, the strategies, processes and measures--of the parent software is specialized in its progeny to the particular application of interest. This happens at time without any other change to the system's modeling or model execution abilities of the software. In spite of this, a new language or simulator is often claimed to have been born.

We are of the opinion, however, that this sharp growth in the number of simulation software packages is unnecessary and in fact harmful to the art and science of simulation. Although such growth does show that the interest in simulation has been sustained (if not showing signs of growth), we believe the proliferation of software hinders the development of a unified philosophy and methodology for simulation which should be the goal of simulation research personnel (c.f., Deutsch, 1976, and Solomon, 1980). In addition to the pedagogical arguments, this proliferation often results in poor familiarity on the part of practicioners with the software that may be useful, and therefore in lower productivity when research is needed to identify and become expert in the use of the software that best fits the simulation project (c.f., Edgecomb, 1976).

However, because this over-abundance of simulation software is partially due to the specialization of performance measurement approaches to production and criminal justice systems, marketing problems, and the like, it is obvious that a philosophy of performance measurement like that discussed by Deutsch (1976) can assist in the development of software specialized not to particular applications but to the

type of measurement approaches required. Furthermore, if new simulation languages or simulators are developed around the axioms of a comprehensive measurement philosophy, then certainly this software would be a significant contribution rated as a true innovation in simulation software design. If, on the other hand, existing software can be adapted to a more universal measurement approach, then the magnitude of the changes required would undoubtedly be regarded as a considerable, if not an innovative, step in the advancement in simulation software. Of course the dissemination of the measurement-oriented simulation software would in either case help to control the growth of applications-oriented software, and this we have already said would be advantageous.

In the remainder of this paper, therefore, we examine the measurement inadequacies of simulation software within the context of the theory of performance measurement. By approaching the problem in this manner, we are able to suggest specific guidelines for the practice of simulation projects having a performance measurement emphasis, and this apperception is then used to provide a general structure for the performance measurement function in simulation software.

IV. TWO PERFORMANCE MEASUREMENT ORIENTATIONS

Our purpose in highlighting the measurement inadequacies of simulation software is, first, to identify the principal omissions that are to be found in existing simulation languages and simulators, and, second, to demonstrate how performance measurement theory can be used to restructure this and future software. Making it easier to use simulation software for measurement studies, if the second objective can be met, then the proliferation of application-specialized software packages should greatly diminish.

We of course have just asserted that the proliferation of simulation software has partially been motivated by the host of applications for which the simulation paradigm has been chosen; yet, we have not described the form of the specialization that is imposed, other than to suggest that it affects the software's ability to measure the performance of similar systems whose measurement philosophies differ from that of the original design. We will now show that the specialization of simulation software to particular applications has not of itself been the most important deterrent to the widespread use of the more general simulation languages and simulators: The concomitant specialization of the measurement approaches to specific model behaviors has also had an important limiting effect. However, we will also show that the specialization of performance measurement to particular model behaviors is not the only limitation to be found today, and so we will explain the importance of the other inputs to the measurement project shown in Figure 1. We thereby demonstrate that a conscientious effort must have been made to ensure measurement flexibility--that is, for accomodating diverse measurement goals, model behaviors and information utilities--for truly general-purpose simulation software.

IV.A. Dimensions of Software Specialization

In our discussion of the specialization that is evident in simulation software, we shall only consider that occurring in the following three areas:

1. in the system's model,
2. in the measurement approach--tailored to the existing conceptual model of a system, and
3. in the measurement approach--tailored to a (reasonable) mix of meas remea- o-

We have already seen that the conceptual model of the class of systems which can be accomodated by general programming languages is nonexistent, but that a conceptual model becomes more-and-more precisely defined as one changes to simulation languages and then to simulators. The increased precision in this conceptual model is what we shall refer to as the specialization of the system's (conceptual) model, and we will use the specialization in the system's model as a benchmark to guage the degree of specialization of the measurement approach because the latter can also be considered to be the product of a "measurement model." Our ultimate objective will be to compare and contrast the degree of specialization of two simulation measurement models in order to evaluate the future needs of more general measurement approaches.

We distinguish between the above two orientations of the measurement approach in simulation to define the characteristics of two quite different measurement models. We shall refer to the second emphasis as specialization to the Simulation Measurement Process, where we define the SMP as a measurement-oriented description of the stages of the simulation project (see Table 2; c.f., Shannon, 1975). Simulation software specialization to the SMP is differentiated from measurement's specialization to the attributes of the system's model and its behavior, because each is the product of a software developer's attempts to achieve two quite different goals. On the one hand, the measurement approach (strategies, processes and measures) is oriented toward a particular simulation model (as for a simulator) or toward a particular set of model behaviors (as for a simulation language); on the other hand, the emphasis to the SMP conveys the fact that a software designer has tried to facilitate the measurement of simulated behaviors for those

measurement goals, model behaviors and information utilities which can reasonable be expected in simulation studies based on knowledge of the Simulation Measurement Process.

IV.B. Specialization to the System's Model

The differences that we see between the two specializations of performance measurement in simulation software translates into two very different philosophies of software development. The specialization to the system's model has historically been the perspective subsumed. Whenever new software has been developed, either instruction sets for constructing measures of performance have been designed for users of simulation languages or vectors of performance measures have been developed to provide a measurement capability to users of simulators. In either case, the measurement goals are explicitly linked to particular kinds of model behaviors. For example, many simulation languages and simulators are capable of modeling waiting lines, but their emphasis on performance measurement is on those kinds of models, notably of production processes, which are stable and whose measurement data is essentially constant. Therefore, the performance of such systems can be adequately summarized by a sample mean (and it usually is). Either the mean number of parts waiting to be repaired or the mean time it takes automobiles to move between stations on an assembly line, would be determined in many simulators by simply referencing the name of the variable; their means would automatically be computed for the report of each simulation run.

Most actual manufacturing systems must be stable over the planning horizon used to evaluate different production configurations, or else they are too expensive to maintain, say, at alternately high and then low

production rates. However, there exists many other simulation applications which are interested in simulating longer planning horizons and therefore nonstationary behavior as when a significant change in the sample mean over time becomes germane.

Criminal court models are one example of systems which require measures of performance that capture temporal changes in the system's behavior when their planning horizons are on the order of a couple years or more. In actual court systems, many measures of performance have recently been shown to change dramatically from year to year (see Hindelang, et al., 1977), and Criminal Justice researchers would undoubtedly wish to model such change in order to ensure that what is being represented by the model approximates that actually observed. Thus, measures of performance must be formulated for validating the modeled behavior. One scheme for accomplishing this is to compare identical measures calculated using the simulated system's output with those from the actual system's measurement date. If it were not for the limited measurement perspective of most simulation software--their abilities to model a court system are often quite adequate--then a great deal more software could be used to simulate and measure the output of a simulated court model than is now the case; that is, without making substantive additions or changes to the software package.

As it happens now, a complex simulation model often requires that additional software be developed in a high level programming language to augment the measurement approach of the original simulation software. This is a result of the specialization of the measurement approach to a rather limited class of system's models for which the software was originally intended; the software is nevertheless used today to simulate more complex phenomena which the software's conceptual

system's model can accomodate but for which the measurement approach is totally inadequate. The approach of some simulation project teams is to append the needed measurement software to existing software, while others choose to develop their own entirely new software in spite of the former alternative's greater economy.

But the design of simulation software, solely for the purpose of overcoming the measurement inadequacies of existing software by specializing its measurement approach to the particular system being modeled, repeats the same mistake that drove the project team from the established simulation languages and simulators. Moreover, it is this kind of response of simulation project teams that we generally oppose on the grounds that it perpetuates the development of new simulation software motivated more by reasons (3) through (7) in Table 1 than by the perception of an opportunity or need for software innovation.

Let us now consider an approach to performance measurement in simulation which better meets the needs of simulation practitioners. Earlier, we viewed the emphasis of the measurement approach on the Simulation Measurement Process as another dimension to the specialization of simulation software. We now show what this means for simulation software.

IV.C. Specialization to the Simulation Measurement Process

In reference to the specification of the Simulation Measurement Process (SMP) in Table 2, only phases six through ten are truly measurement oriented and therefore require measurement approaches designed in earlier phases consisting of measurement strategies, processes and performance measures. Ideally, one would determine the measurement goals, model behaviors and utilities for information during phase one before outlining each measurement strategy, process and measures for every one

of the measurement approaches required at each phase. The fact that the resulting measurement approaches will often differ from one another at each phase can be demonstrated by exploring the effects that the various inputs to the measurement project (Figure 1) have on the processes, measures and strategies of the measurement approaches. We wish to show that this is true with a simple example, thus allowing the reader to infer for himself that our conclusions based on the one example would be generalizeable to the other measurement phases of the SMP and to other measurement project inputs. Our example is of a criminal court simulation model, and our comparison of measurement approaches is for the model initialization and validation phases (Phases 7 and 8 in Table 2).

The initialization of a trial court model may be concerned with, among other things, the selection of representative initial values for the number of offenders waiting for their trials (see Richards and Deutsch, 1978). The behavior of initializing systems and the utilities of certain kinds of information and performance measures for initializing systems are discussed by Richards (1980). The strategy for initializing the model might require, for example, that the most representative operating conditions are to be found for the court model on a preliminary run and that these conditions be used as the starting values for the simulation model. The appropriate measure of performance might be either the sample mean or the sample mode (most likely value) of the distribution of the number of offenders waiting to go to trial. The strategy required to calculate the mean would be either 100% inspection (e.g., the number of offenders waiting for trial is determined every simulated day) or some random sampling plan in which simulated days are picked according to some scheme to simultaneously minimize the

cost of data collection against the variance of the sample mean (see Cochran, 1977). The measurement process would then be the calculation of a simple average in the one case, or it would entail the construction of a frequency histogram and then the picking of the most likely value from the histogram.

The goal of model validity, on the other hand, should require that the actual number of persons found to be waiting for trial be comparatively close to that obtained in simulation runs. Rather than computing the average number of offenders as might be required to initialize the model, other measures of performance derived from the same data may be more meaningful for testing the validity of the simulation model. For example, measures that make a direct comparison between the simulated data and the data collected from the modeled system would be preferred by measurement strategists; such relative measures might include some of the following:

1. the percentage deviation between the actual and the simulation-produced average number of waiting offenders,
2. the ratio of the actual and simulation-produced averages,
3. the ratio of the actual mean to the most likely value (mode) produced by the simulation, or
4. the sum of squared deviations between the actual average and the time series of the number of offenders awaiting trial that is recorded during the simulation run.

The measurement strategies and processes required to compute these validation measures do differ from one another, if only slightly. In all cases, the measurement strategy would require that the actual average number of offenders be provided, and in cases (1) - (3), the

remainder of the measurement strategy would be similar to those suggested for the initialization phase. The measurement processes for cases (1) - (3) would also require simple calculations and, in addition, the sample mode would need to be determined for case (3). For case (4), however, the measurement strategy would require a mechanism for creating and storing a time series, and such factors as how often to collect each observation and what time interval to leave between each data point are essential to the specification of this measurement strategy; the measurement process then simply requires a few elementary calculations to determine the performance measure.

The conclusion to be made from this example is that the measures of performance, the measurement strategies and the measurement processes not only may differ from one another during each phase, but that they will also differ--and often the strategies and processes will differ more dramatically than shown here--from one phase to the next. The need for software specialization to the Simulation Measurement Process, rather than to the system's conceptual model, is demonstrated by both of these differences. The measurement goals, the system behaviors and the utilities for information--the so-called inputs to the measurement project--are not the same for the initialization and the validation phases of our example even though the data used throughout was the same: the number of offenders awaiting trial. In the first place, the measurement goal is to find the most representative values for the measurement data; the system behavior usually found resembles the time series of the level of water in a dam being filled for the first time. Moreover, the utilities for information about the system favor those measures that describe the number of offenders which

recurrs in the court model most often (e.g., the mode). For the validation example, the measurement goal is to make a comparison between actual and simulated system behavior; the system behavior, which we have not exactly specified, would nearly exactly define what type of measure of performance, strategy and process would be most useful. The utilities for information obviously favoring relative measures over others would, of course, further restrict the process, measure, strategy options available.

Because of the ever-changing goals, behaviors and utilities that present themselves in simulation projects, then, we conclude that software specialized according to the dictates of the Simulation Measurement Process and a reasonable set of appropriate performance measurement project imputs should be well equipped to handle the measurement requirements of many simulation projects.

V. CURRENT MEASUREMENT INADEQUACIES

We now support our claim that existing simulation software are deficient for measuring the performance of general-application system's models. We show that the emphasis in software development has been on a rather limited class of behaviors that may result from rather simple system's models instead of on the measurement goals, system behaviors and information utilities that are required inputs for the application of the Simulation Measurement Process to general system's models.

We believe to be particularly important in this appraisal the appearance in the software of any prior commitments to measurement processes, measures or strategies in the form of programmed structures that produce specific measures of performance (measures, recall, are

the products of measurement approaches.) The programmed structure is to be contrasted here with any ability to measure the performance of a system's model, which delegates the responsibilities for designing and implementing the required measurement approaches to the wisdom, the imagination, and the modeling and programming skills of software users. Our inclination is to believe that the possession of such an ability to implement measurement approaches in a software package is nearly as important a limitation as no measurement ability at all, because of the variability in the needed skills between simulation project teams. Therefore, we shall not consider an ability to develop measurement approaches a prior commitment to performance measurement, just as we would not consider must high level programming languages to be so inclined even though they obviously can be used to develop measurement approaches.

V.A. The Specialization Hypothesis

As we said earlier, the proliferation of simulation software can be attributed in part to the inadequate treatment of performance measurement problems in general purpose simulation languages and simulators. Our contention is that the limitations of simulation software can be explained by the existence of inferior measurement approaches (processes, measures and strategies) or by the lack of an adequate prior commitment to generally useful measurement approaches in simulation software. In addition, we further contend that one key to understanding their present measurement inadequacies lies in the historical emphasis that has been placed on the specialization of performance measurement to the system's model--rather than to the Simulation Measurement Process.

This behavioral orientation seems to have prevented software developers from addressing the wide range of measurement problems that face practicioners during the course of a simulation project, and this has in turn forced the simulation project team to bring to its ranks someone skilled in a general purpose programming language. His function is either

1) to develop entirely new simulation software specially designed for the project or the particular application, or

2) to augment the existing software with a more useful measurement approach.

Our position is that these expenses would be unnecessary much more often than they are now if the measurement approaches of both simulation languages and the more general simulators had been adapted to the requirements of the SMP rather than to a limited class of system's models' behaviors. To illustrate the hypothesis that simulation software in the past has emphasized too heavily the measurement of rather specific system's model behaviors at the expense of other more general behaviors, measurement goals and information utilities--this is in fact the distinction to be made between specialization to the system's model versus specialization to the SMP--the three dimensions of software specialization are drawn in Figure 2.

Arbitrary scales in Figure 2 depict the degree of specialization present: a "10" implies the greatest amount of specialization. The two boxes shown in the figure are drawn with solid lines to present a conceptual model of the combinations of specializations for which existing simulation languages and simulators are to be found. We of

course already know that simulators are more specialized in their system's models than are simulation languages and the figure arbitrarily shows this at degrees 10 and 5, respectively. As for the orientation of the measurement approach to the system's model, we assume that the degrees are--again arbitrarily--10 and 5, to reflect the greater opportunity for orienting the measurement model to the system's model when the system's model is itself more specialized. We complete our discussion of the hypothesized scope of simulation software by noting that we have selected the levels of specialization to the SMP to be 4 and 3 to once again reflect both the different opportunities for such specialization as well as the incomplete development along this dimension.

We shall propose in a later section how these two boxes may be extended along the third axis depicting the specialization of the measurement approach to the SMP (see dashed lines, Figure 2). Note that the boxes have not been simultaneously shrunk along the axis indicating measurement emphasizing the system's model, because this kind of emphasis is subsumed on the more general third axis; the converse is obviously not true, however.

We now propose to illustrate the measurement deficiencies of existing software.

V.B. The Evidence

The degree of the prior commitment to performance measurement in simulation software is actually quite poor as demonstrated by the lack of variety both in the performance measures which can be generated and in the manner in which they are generated (viz, the measurement strategies and processes). This limited assortment of performance measures and of the measurement strategies and processes suggests a poor commitment to a general performance measurement theory in simulation software.

Consider the following items:

1. In general purpose programming languages like FORTRAN or APL (see Sammet, 1972), the prior commitment to measurement is nil: there does not exist any conceptualization or facility for any of the measurement processes, measures or strategies which may prove useful in simulation projects.

2. In simulation languages like GPSS, SIMSCRIPT or SIMULA (see Fishman, 1973), the scope of measurement-related activities is limited to the calculation of a few simple functions of nearly any variable which the system's model has generated during a simulation run (this is the measurement process); however, data collection (also part of the measurement process) is implemented with little or no flexibility provided in the choice of sampling plans or of other important elements of a measurement strategy.

3. The scope of measurement approaches is yet more limited than it is in simulation languages in existing simulators; excellent examples of such criticism include the general purpose network simulator GNS (see Hogg, 1975) or many of the application-specialized packages like the criminal justice simulator JUSSIM (see Chaiken, et al., 1975). A brief menu of performance measures is usually provided to the user for his determination of those that are to be calculated and output with the reports that accompany each run of a model. This menu is confined to those measures which can be calculated by a few functions of a small subset of the variables that the simulator generates during each run, and the data is also collected with little regard for the many choices that must be made in developing a measurement strategy for more general systems.

The structure of measurement in simulation languages centers around an ability to calculate simple functions of data automatically generated by the simulation model structure during the course of a simulation run. Because of the simplicity of the measurement approach, performance measures which result from complex sampling plans or from the estimation of parameters of statistical models (the one is a common consideration in measurement strategy formulation; the other is a frequently used measurement process for estimating measures of performance for dynamic systems as well as many others) cannot be had. One is led to believe that the measurement approaches are either model- or behavior-oriented because they are severely limited by the types of measures which can be produced by simple functions. The undeniable conclusion is that a much narrower perspective has been assumed for most simulation languages in their definition of the relevant system behaviors, measurement goals and information utilities, that should have occurred with the knowledge of the Simulation Measurement Process and its application to more general measurement problems and system's models. We therefore feel justified in our assessment portrayed in Figure 2 of the degree of measurement specialization in simulation languages to the system's model (behaviors) rather than to the SMP.

The great flexibility of unstructured measurement approaches belonging to simulation languages is not to be misconstrued, on the other hand, as a positive sign of any prior measurement commitment to the SMP. For example, the unstructured measurement abilities of simulation languages can be assumed to focus on their modeling flexibility and on their ability to have program appendages written in a general purpose programming language. Unusual sampling plans can be accomodated in simulation languages for instance by combining measures of performance using

either the flexibility of the language or the appended program features. Returning to our earlier court example, we need to determine the average number of offenders waiting to go to trial. By separately tabulating those offenders who commit different crimes, a special sampling plan can be implemented using a language's modeling flexibility alone to reduce the variance of the average number of offenders who wait for a trial, regardless of their crime. (This reduction in variance is predicated on the choice of performance measure, and the reader is advised to refer to Cochran, 1977, for an explanation of stratefied sampling plans.)

Although the ability to accomodate more unusual measurement approaches exists for the sophisticated system's models or for the programmer who takes the time to learn how to attach a general purpose programming language appendage to a simulation language, the prior commitment to performance measurement is not enhanced by these efforts. In fact, they merely increase the software development costs of simulation projects, and they help to explain the proliferation of new specialized simulation software for those reason given in Table 1.

We turn now to the measurement inadequacies of system simulators. Because of the normally small size of their menus of performance measures, simulators provide even fewer measures of performance to a simulation project team than would a simulation language, while those measures that are provided are usually tailored to an even narrower system's conceptual model. This is the case, in part, for the same reason that the system's model is itself quite limited: the programming effort required to generalize a simulator can be considerable and is therefore not usually attempted. In addition, the measures of performance computed are most often the means and standard deviations of particular

data types--like the number of entities (or offenders) delayed by a queue (or a court). As we have shown in our earlier discussions of the measurement orientations to the system's model versus the SMP, this kind of restriction ignores the other goals of measurement which one would find in any application of the SMP. Those goals related to model initialization and validation that we have illustrated earlier, tend, for example, to be given less attention under such circumstances.

Although the existing structures for performance measurement are quite inflexible in simulators, the ability to develop measurement approaches is again hampered either by a project team's lack of expertise in the use of the simulator's modeling ability and in their ability to append high level program units to the simulator to implement the desired measurement approaches, or by the size of their budget so that such changes may be precluded on the basis of cost. To be sure, the modeling flexibility is usually far less for simulators than for simulation languages, requiring greater programming skills than modeling expertise to specialize further the measurement approach of a simulator, but this flexibility is nearly always available at a price to the project team just as it is for simulation languages. However, specializing existing simulation software to every project that comes to a simulation team is both impractical and terribly expensive; furthermore, as we have since shown that the measurement inadequacies of existing software can be overcome by software specialized to the Simulation Measurement Process rather than to a particular system's model, it should be replaced either:

1) with completely new packages whose measurement approaches are based on the SMP applied with a greater appreciation for the

inputs to the measurement project, or

2) with newer versions of the package in which the particular measurement applications are subjugated to the needs of the SMP as we have outlined them elsewhere.

We summarize the aformentioned measurement inadequacies, their symptoms and their more obvious solutions in Table 3. As with the simulation languages, however, the design and development of new simulators to re-structure the measurement approaches of existing simulation software should at least accomplish the following. It should reduce the costs incurred for each simulation project to specialize existing software to the one application of the Simulation Measurement Process; it should reduce the need to develop specialized simulation languages and simulators for each application area that exists; it should reduce the needless specialization of new software to the measurement approaches of particular system's models; it should make project teams more aware of the program structures required to implement general approaches in concert with the needs and objectives of the Simulation Measurement Process in the event that a particular application can justify specialized software on the grounds of reason (1) in Table 1; and it should expand the types of measurement strategies, processes and measures that are readily available for simulation measurement projects.

VI. MEASUREMENT THEORY FOR SOFTWARE DESIGNERS

Throughout this paper we have tried to show how simulation software, as one tool for the measurement of the performance of system's models, is currently not well structured for this purpose. We have illustrated some of the symptoms of this problem, but the evidence supports the conclusion that the measures of performance that may be readily produced

by the software--that is, without additional programming or artful reformulation of the system's model--are not alway appropriate for the array of measurement goals, system behaviors or information utilities that are to be found in most simulation measurement projects. The recurring theme of our investigation has been that simulation software has been developed with a bias toward measuring the behavior of particular systems and their applications and that even the host of behaviors to be expected through the entire simulation measurement project using the same model has not been taken into account while designing the measurement approaches of even commercially available simulation packages.

Our calling for the specialization of simulation software to the Simulation Measurement Process has been one way to indicate some of the areas for which measurement has been neglected, where again the difficulty for the simulation project team surfaces as a need for a particular measure of performance which cannot be satisfied by the system's model or by the existing measurement strategies and processes available to them. To correct for the missing measurement structures, however, researching every possible application (i.e., every measurement goal, system behavior and information utility) which could be addressed by a software package's system's model is not a practical approach for resolving the measurement inadequacies of existing simulation software; the task would be impossible to define much less pursue. Therefore, what is needed is a conceptual basis from which the software specialist can work to enable the user to quickly construct the measures of performance which he deems useful, with as few prior restrictions having been imposed by the software designer's appraisal of the applicability of particular system behaviors, measurement goals or information

utilities to the measures, measurement strategies and measurement processes which the user may require.

This suggestion perhaps sounds like it conflicts with our earlier call for the specialization of simulation software to the SMP and its inputs, and we now clarify this point.

VI.A. A Conceptual Model for Measurement

If we were to examine the computer codes of existing simulation software, we might determine that the measurement approach consists of the following steps:

1. define the measures of performance,
2. specify the data to be collected and how it is to be collected,
3. collect this data during the simulation runs, and
4. compute the measures of performance.

According to our earlier definitions, item one specifies the measure of performance, two is the delineation of the measurement strategy, and items three and four loosely define the measurement process that now exists for simulation software.

If it were possible to encode more general measurement strategies and processes and performance measures in the software, and at the same time ensure that most of the Simulation Measurement Process was incorporated therein with a variety of input combinations, then we could essentially consider that the software had been specialized to the SMP in a general way (that is, for general goals, behaviors and utilities). The key would be for simulation software specialists to determine a rich combination of measurement strategies, processes and measures which can be most usefully implemented in their software packages for measuring system performance for the types of goals, behaviors and information utilities which might be expected in actual

projects, so that anyone who uses the package can then pick the combination of measurement strategies, processes and measures which best suits the particular goals, behaviors and utilities that he has already identified for his projects. This is one area where the measurement strategy, process and measure selection models of Deutsch (1976) and later of Deutsch and Malmborg (1980) can be of use both in the design of new simulation software and in the formulation of desired measurement approaches as part of the Simulation Measurement Process.

Ideally, the additional measurement structure to be developed for simulation software should give the user the freedom to choose the measures and every detail of both the measurement strategies and the measurement processes in keeping with his expectations for the measurement goals of each phase of the SMP, for the system behaviors to be found at each phase, and for the types of information that would be of greatest value during each phase of his project. Unfortunately, though, this also will generally not be practicable, and we must state that specialization to the SMP can therefore only be obtained for particular applications --i.e., a precise set of behaviors compiled with exact measurement strategies and processes. But this approach is not any good either, because it contributes to the proliferation of simulation software, unless such specialization can otherwise be justified as an innovative application as defined in reason one of Table 1.

The simulation software development specialist is consequently forced to define an array of system behaviors, measurement goals and information utilities with which to structure the measurement approaches of simulation software. We recommend that the research on this problem be centered around the phases of the SMP for defining these measurement

inputs, and this will in turn yield specific recommendations for measurement strategies, processes and measures which are directed at these fairly general measurement inputs rather than at the particular behaviors of a single application.

Consider for example some of the goals, behaviors and utilities that we have illustrated throughout our discussions with the court model. The initialization goal had a high utility for the level of the output, and, assuming that the data had a constant expected value, we suggested the sample mean be used as the measure of performance. However, if the mean level had been changing over time, then perhaps a more appropriate measure would have been the expected value of an appropriately increasing function, assuming that the mean number of offenders increased through time. In this case we might prefer (i.e., information utility is matched with the model behavior and the measurement goal) estimating the straight line

$$E(x) = a + bt ,$$

where $E(x)$ is the expected number of offenders after t simulated time periods. This measure contrasts with that for the other model behavior, even though the goals and information utilities are the same. Thus, whereas the measures of performance may differ, they can be predicted before hand and as a result encoded in the general-application software to which we aspire.

It is of course up to the discretion of the individual simulation software specialist, however, who desires to develop generally useful packages with specialized measurement features oriented to the SMP, how far he chooses to carry such specialization. He should, nevertheless, only under those circumstances outlined in reason (1) of Table 1

specialize to a particular application else he impose unnecessary constraints on a user's ability to easily mix measures, processes and strategies to suit his needs.

VI.B. A Performance Measurement Structure

To facilitate the development of the software changes needed to implement more flexible measurement approaches for simulation, we offer a functional description of simulation software which highlights a Measurement Model that encompasses all operational aspects of the measurement strategies and processes designed in the software. See Table 4.

We find the following major tasks to be performed in simulation:

- Monitoring the software's execution (performed by an Executive Model),
- Receiving and validating the input from the user (the Input Model),
- Executing a computer-coded replica of a system (the System's Model),
- Executing the measurement approaches (the Measurement Model),
- Analyzing the results of measurement (the Performance Analysis Model),
- Documenting the results of the computer runs (the Report Model), and
- Executing an experimental design strategy like those described by Box, Hunter and Hunter (1978) for determining if any change in the System's Model effects changes in the measures of performance (the Design/Optimization Model).

These tasks are easily recognized as being consistent with the SMP. This obviously is our goal, but the purpose of differentiating these

functions is to convey a greater sense of the importance of performance measurement as the intermediary between the system's Model and the Performance Analysis and Report Models. See Figure 3.

It is our opinion that the Report, the Performance Analysis, and the Measurement Models have often been confused as serving the same purpose, but it is our hope that these distinctions have clarified the role of performance measurement in simulation software. In addition, it should further simplify the necessary software advancements and, therefore, provide an incentive for software design specialists to tailor new and existing simulation packages to the processes, measures and strategies resulting from carefully analyzed measurement goals, system behaviors and information utilities that arise during the progress of the Simulation Measurement Process.

VII. CONCLUSIONS AND RECOMMENDATIONS

We have tried to show that the proliferation of simulation software today is in part due to the over-reliance of measurement models on the particular characteristics of the systems being modeled or on the conceptualization of a class of systems expected to be modeled. However, we have also tried to show that this abundance of software has in part attempted to overcome the inadequate availability of performance measures in the software which seems predicated on the limited scope of their measurement approaches--i.e., in the breadth of the measurement goals, system behaviors and utilities for information to which they are suited. It is the pre-programmed nature of these inputs which we find fault in existing simulation software. The measurement approaches of existing simulation software must be broadened in accordance with

the Simulation Measurement Process to accomodate the various applications which require simulation treatment throughout the entirety of a simulation project and its ever-changing measurement needs.

We have also during our discourse identified how a measurement model is to be distinguished from report, performance analysis, system's modeling, and experimental design functions in simulation software. This distinction has, we believe, enhanced our ability for structuring new measurement approaches characterized by a user's being able to select an appropriate mix of measurement processes, strategies and performance measures for addressing the measurement goals, system behaviors and information utilities at hand.

What remains to be done is to define the mix of processes, measures and strategies which should be implemented in existing simulation languages and simulators for accomodating the many applications and the several phases of the simulation measurement project that we have outline. To do this, one may use the methodologies of Deutsch (1976) to select the appropriate mixes based on previous simulation projects, or one can base his selection on the methodologies of Deutsch and Malmborg (1980) based on a utility-maximizing, information theoretic approach to picking measures of performance. In either case, however, the development is to address generic system behaviors--not applications or particular models as has been the case in the past.

Finally, before closing, it should be noted that we have left as an open research question whether the measurement model thus envisioned should be an appendage to existing software or whether it should be used to structure an entirely new measurement-oriented simulation language or simulator. In a companion paper by us (1980), we describe the first

attempt at either effort; the requirements of a Measurement Model for the general network simulator GNS are there discussed.

REFERENCES

1. Annino, J. S. and E. C. Russell (1979). "The Ten Most Frequent Causes of Simulation Analysis Failure--And How to Avoid Them!" Simulation, pp. 137-140.

2. Box, G. E. P., W. G. Hunter, Jr. and J. S. Hunter (1978). Statistics for Experimenters: An Introduction to Design, Data Analysis and Model Building. New York: Wiley Interscience.

3. Chaiken, J., T. Crabill, L. Holliday, D. Jaquette, M. Lawless and E. Quade (1975). Criminial Justice Models: An Overview. RAND Report R-1859-DOJ.

4. Cochran, W. G. (1977). Sampling Techniques. New York: John Wiley & Sons.

5. Deutsch, S. J. (1976). "A Conceptual Basis for Effectiveness Measurement of Law Enforcement Activities," in: Performance Measurement and the Criminal Justice System: Four Conceptual Approaches. LEAA/NILECJ, U.S. Department of Justice, Washington, D.C.

6. Deutsch, S. J. and C. J. Malmborg (1980). "A Synthesis of the Matrix Methodology in Performance Measurement," working paper, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, Ga.

7. Deutsch, S. J. and J. E. Richards (1978). "An Evaluation of CJS Models: Simulation Approaches and Recommendations," Proceedings, Pittsburgh Conference on Modeling and Simulation.

8. Deutsch, S. J. and J. E. Richards (1979a). "Planning Socially Optimum Plea Bargaining," ISyE Report Series No. J-79-36, Georgia Institute of Technology, Atlanta, Ga.

9. Deutsch, S. J. and J. E. Richards (1979b). "The Compendium of Performance Measurement Literature," Technical Note 4, Department of Justice, LEAA Grant No. 78-NI-AX-0003.

10. Deutsch, S. J. and J. E. Richards (1979c). "Criminal Justice Simulation: Trade-Offs Between Performance Measurement and Other Model Design Criteria," Technical Note 5, Department of Justice, LEAA Grant No. 78-NI-AX-0003.

11. Deutsch, S. J. and J. E. Richards (1979d). "A Generalized Network Simulation Model of the CJS: User's Guide," working paper, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, Ga.

12. Deutsch, S. J. and J. E. Richards (1980a). "A Generalized Network Model of the Criminal Justice System--Programmer's Guide," ISyE Report Series No. J-80-08, Georgia Institute of Technology, Atlanta, Ga.

13. Deutsch, S. J. and J. E. Richards (1980b). "A Generalized Network Model of the Criminal Justice System--A Prescription for a Performance Measurement Model," ISyE Report Series No. J-80-07, Georgia Institute of Technology, Atlanta, Ga.

14. Deutsch, S. J. and J. E. Richards (1980c). "Formulating a Measurement Model for Simulation," Technical Note 10, Department of Justice, LEAA Grant No. 78-NI-AX-0003.

15. Edgecomb, G. D. (1976). "Validation - The Bottleneck in System Simulation," Simuletter, Vol. 8, No. 1, pp. 40-44.

16. Fishman, G. S. (1973). Concepts and Methods in Discrete Event Digital Simulation, New York: Wiley Interscience.

17. Hindelang, M. J., M. R. Gottfredson, C. S. Dunn and N. Parisi (1977). Sourcebook of Criminal Justice Statistics- 1976. LEAA, U.S. Department of Justice, Washington, D.C.

18. Hogg, G. L. (1975). GNS User's Manual. National Clearinghouse for Criminal Justice Planning and Architecture, University of Illinois, Champaign-Urbana, Ill.

19. Kleijnen, J. P. C. (1975). Statistical Techniques in Simulation, Parts I and II. New York: Marcel Dekker.

20. Kotler, P. (1971). Marketing Decision Making: A Model Building Approach. New York: Holt, Rinehart and Winston.

21. Phillips, D. T., P. Piumsomboom, G. L. Hogg, R. J. Heistenberg, M. Handwerker, S. Sathaye, V. J. Dharia and R. Goforth (1979). "GEMS: A Generalized Manufacturing Simulator," Proceedings of the 7th NSF Grantee's Conference on Production Research and Technology, Ithaca, N.Y.

22. Richards, J. E. (1980). "The Initialization Problem in Simulation," working paper, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, Ga.

23. Richards, J. E. and S. J. Deutsch (1978). Planning Optimal Plea Bargaining and Sentencing Strategies for a State Judiciary, technical report, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, Ga.

24. Sammet, J. E. (1972). "Programming Languages: History and Future," Communications of the ACM, Vol. 15, pp. 601-610.

25. Shannon, R. E. (1975). Systems Simulation, The Art and Science, Englewood Cliffs, N. J.: Prentice Hall.

26. Solomon, S. L. (1980). "Building Modelers: Teaching the Art of Simulation," Interfaces, Vol. 10, No. 2, pp. 65-72.

TABLE 1

Reasons for the Proliferation of High Level Programming Languages
(source: Sammet, 1972; pp. 602-603)

1. A really new language concept has been developed and/or a new application area is deemed worthy of having its own language.

2. After experience with a particular language, its deficiencies are clear enough that a complete new language is needed--and justifiably created--to correct them.

3. Facilities of several languages are best combined into a single new language.

4. It is felt to be easier to get additional capability or changes in style with a new language rather than to extend or modify an existing one.

5. It is fun to design and implement a new language, and someone wants to do it and can obtain the funds.

6. There are personal preferences and prejudices against the existing languages even though one of these languages might serve the purpose for which the new language is intended.

7. The developer is unaware of the existence of a language that meets his needs, so he creates his own while believing he is meeting the conditions of (1) or (2).

TABLE 2

The Simulation Measurement Process (SMP)

Phase 1. Formulate the Measurement Strategies— Clarify the measurement goals to be achieved. Define the model behaviors expected to be of interest during the measurement Phases 7-10, the utilities for particular kinds of information that may result, and the role of simulation in the overall measurement strategy. Specify the objectives of each phase of the measurement process and show how they best qualify in helping to attain the measurement goals.

Phase 2. Define the Measurement Processes— Select the simulation language that best serves the measurement goals. Focus the steps of the measurement processes used in the following phases to best achieve the measurement goals of each phase and the Simulation Measurement Process as a whole. Refine the measurement strategies whenever necessary.

Phase 3. Formulate and Implement the System's Model— This phase includes the formulation and translation of the system's model either to a set of instructions if a simulation language is used, or to data for input to a simulator. The preparation of input data and parameters and the verification that the computer model works as intended are also necessary.

Phase 4. Formulate and Implement the Measurement Approach— Define the measures of performance desired and specify the processes required to derive estimates for them, being consistent with the measurement strategies but within the limits imposed by the software. Verify that their implementation is correct. Prepare any necessary input data.

Phase 5. Specify the Analyses— Kleijnen (1975) and Fishman (1973) both discuss a variety of analytical techniques that are useful in simulation. The model builder should select those that provide information most suitable to the objectives established in Phase 1.

Phase 6. Decide on the Simulation Output— This requires the selection for output of particular measures of performance and analyses conducted internally by the simulation language, and should be done coincidentally with the formulation of the measurement approaches of Phases 7-10.

Phase 7. Initialize the System's Model— This requires preloading the system if necessary so that the initial states of the simulation model or the initial values of performance measures

(continued)

TABLE 2
(cont'd.)

correspond to desired values (see Fishman, 1973). This phase often requires a unique measurement approach that differs from the other phases because of the existence often of different measurement goals, system behaviors and information utilities.

Phase 8. Validate the System's Model— Compare the states of the system of trial simulation runs with those observed in actual circumstances similar to those portrayed in the simulation model. This phase usually requires a unique measurement approach, different from other project phases.

Phase 9. Validate the Measurement Model— Compare the measures of performance of trial simulation runs with those observed in actual circumstances. See Deutsch (1976) for a discussion and illustration of measure validation and the unique measurement approach often required that is different from the other project phases.

Phase 10. Design and Run Experiments for the Simulation Models— Experimental designs for a variety of purposes may be chosen--e.g., to determine which changes to the system's model significantly affect the vector of measures of system performance, to determine the magnitude of these effects, or to find the magnitudes of the changes needed to yield the optimum performance vector. The measurement approach for this phase is usually different from those used elsewhere in the project.

Phase 11. Communicate the Results-- Document the results, the conclusions and any recommendations for additional measurement.

TABLE 3

Measurement Inadequacies Found in Simulation Software

PROBLEM	SYMPTOMS	SOLUTION
1. Much of today's simulation software has become too specialized to particular applications in their measurement approaches to be generally useful.	a) New simulation languages and simulators of special applications flourish without any particular innovations present or any other tangible justification. b) Even though systems' models possess characteristics easily modeled by general purpose simulation software, measures of performance specialized to particular applications limit the software's usefulness for other applications.	The general purpose measurement approaches of simulation languages and simulators must be made to accommodate more universal measurement goals, system behaviors and information utilities and thus place less emphasis on the particular goals, behaviors and utilities found to exist.
2. The ever-changing measurement goals found in simulation projects are usually not recognized in the measurement approaches of simulation software.	Measurement processes, strategies and most importantly measures of performance are not geared to the particular measurement goals of simulation projects--e.g., validation, initialization and experimentation.	Common strategies, processes and measures must be implemented in simulation software to accommodate the phases of measurement projects in simulation.
3. Limited flexibility is provided for choosing measures consistent with measurement goals, system behaviors and information utilities.	a) For simulation languages, only functions of data can be calculated. b) For simulators, usually only particular measures may be computed. c) Complex statistical functions whose parameters may be useful as measures cannot generally be estimated. d) Sampling plans for collecting data are inflexible--often 100% inspection is imposed, and rarely are any alternative methods provided for calculating estimates to reduce their variances.	More general measurement strategies and processes must be provided to users of simulation software to accommodate more general measurement goals, model behaviors and information utilities.

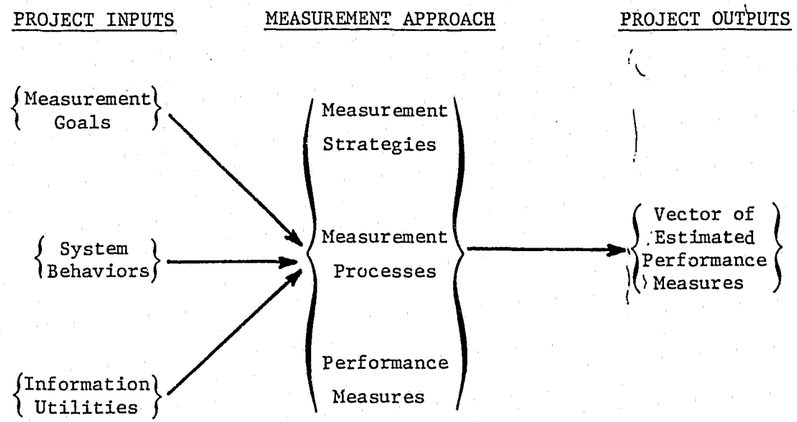

FIGURE 1. The Measurement Project is the Basis for the Practice of Performance Measurement Theory

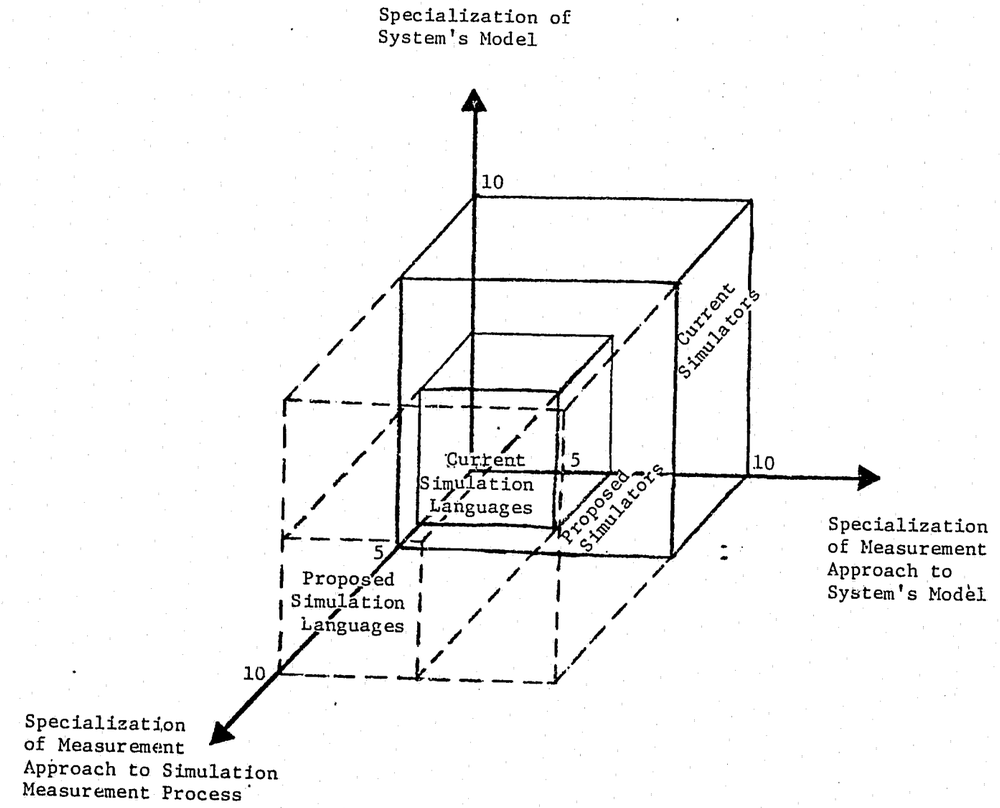

FIGURE 2. The Relative Degree of Specialization in Simulation Languages and Simulators Along Selected Dimensions

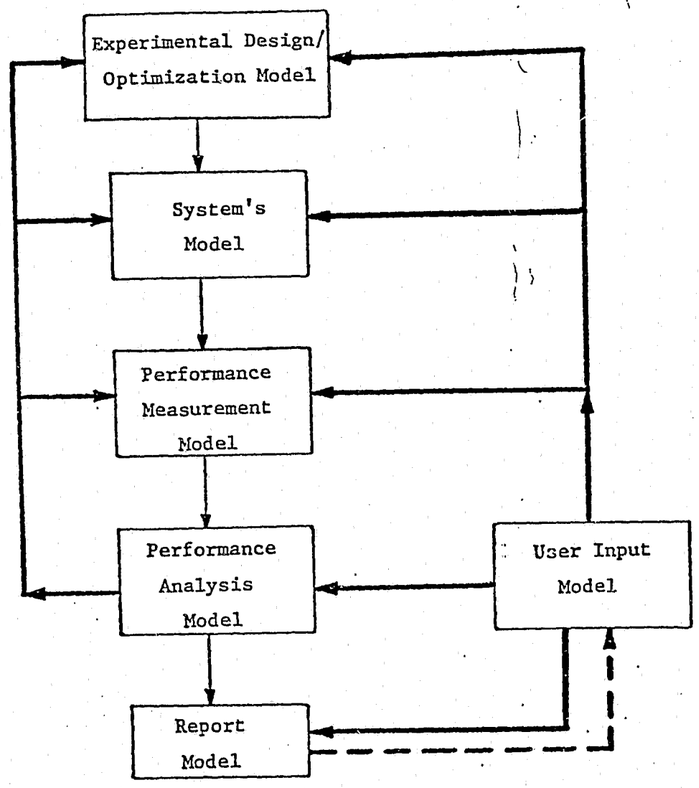

FIGURE 3. A Functional Decomposition of Simulation Software Complete with Information Exchanges

—— = Information exchanges under the control of the Executive Model

– – = Likely places for human intervention in the conduct of the simulation project

Table 4. A Functional Description of Simulation Software Specialized to the Simulation Measurement Process.

1.	Executive Model	It is the master controller for the software, determining the sequence in which the other models are executed and passing parameters and data among them.
2.	Input Model	It receives every model's parameters and input data directly from the user and it transmits them to the other components via the Executive.
3.	System's Model	This is the model of the system's objectives, processes, resources, and environment.
4.	Measurement Model	This model defines, collects data for, computes, and saves the performance measures for the System's Model specified by the user in the Input Model.
5.	Performance Analysis Model	This model analyzes the computed measures of performance to determine what action the Executive Model is to take subsequently. Depending on the phase of the Simulation Measurement Process, the Executive may be instructed to terminate the run and print reports for the user or it may be instructed to modify any of the following models or their parameters: the Measurement Model, the System's Model, and the Experimental Design/Optimization Model.
6.	Report Model	It takes the data collected by the Measurement Model and analyzed by the Analysis Model, and outputs particular reports requested by the user.
7.	Experimental Design/ Optimization Model	This model instructs the Executive to run the System's Model with particular parameter values and System's Model configurations in order to effect changes in performance criteria that ultimately may be optimized.

END