Proceedings of the 1994 Winter Simulation Conference
ed. J. D. Tew, S. Manivannan, D. A. Sadowski, and A. F. Seila

A CONCEPTUAL FRAMEWORK FOR SIMULATION EXPERIMENT DESIGN AND ANALYSIS

Yu-Hui Tao
Barry L. Nelson
Department of Industrial, Welding and Systems Engineering
The Ohio State University
Columbus, Ohio 43210

ABSTRACT

This paper summarizes a conceptual framework for computer-assisted Simulation Experiment Design and Analysis (SEDA). We argue that this SEDA framework is a good one based on its properties. A prototype SEDA CS, which is not described in this paper, is under development for demonstrating the feasibility and appropriateness of this framework.

1 MOTIVATION

Computer simulation is a useful tool for supporting decision making, especially when there is no available analytical method. However, computer simulation is a complex task for many users. Simulation experiment design and analysis (SEDA) is as critical as simulation modeling to obtaining useful results. Our concern is how simulation users can be assisted with SEDA. This paper presents a conceptual framework for SEDA. In order to be concrete we focus on the problem of comparing mean or expected performance of queueing-network systems.

A typical simulation project includes the following stages (Taylor et al. 1988): Formulation, programming, verification, validation, experiment design and analysis, and recommendations. Accordingly, a traditional simulation environment requires four classes of knowledge (O'Keefe 1986): Knowledge about the problem domain, knowledge about simulation modeling, knowledge about the programming language, and knowledge about statistics.

Shannon (1985) estimated that one needs to spend "at least 720 hours of formal classroom instruction plus another 1440 hours of outside study (more than 1 man-year of effort)" to acquire the basic simulation tools. This does not include the extra effort required to gain real-world, practical experience in order to become proficient. Therefore, it is not easy for a novice to perform a thorough simulation study.

A survey of large U.S. firms by Thomas and DaCosta (1979) indicated that only 28 percent of individuals in operations research departments have backgrounds in statistics or mathematics. Another survey by Nelson et al. (1987) showed that computer simulation is the second most used OR tool in production management, after regression analysis. A recent survey by Dyer et al. (1993) ranked computer user skills, probability and statistics, and simulation techniques as the number 1, 2, and 3 technical skills, respectively, for a recent graduate of a master's program, and the number 1, 2, and 4 technical skills, respectively, for a seasoned professional in MS/OR.

The implication of the three survey results is that the lack of understanding of how to correctly design the simulation experiments, and analyze and interpret the simulation output, may be a problem in the simulation user population. Moreover, based on our experience, users with basic statistical background nevertheless may not know how to appropriately apply statistics in simulation. As a result, even though the population of simulation users who have statistics background may increase, the problem still exists. Therefore, a computer system (CS) that can shield the users from the knowledge of statistics in simulation is critically important and useful to the success of a user population without qualified statisticians or adequate knowledge of the SEDA problem solving.

2 OVERVIEW

We are interested in knowing how we can help simulation users with SEDA through a CS. We agree with Freeman (1980) in "The Nature of Design:"

> It is essential that one have a conceptual understanding of a complex activity in order to master its intricacies. Without such a framework, one has only isolated facts and techniques whose interrelationships may be

obscured. Without an understanding of broad classes of phenomena, one is condemned to understand each new instance by itself.

This paper presents a conceptual framework for the complex activities in SEDA, and illustrates the framework with a brief example. The definition of six SEDA components forms the basis of the framework. The framework itself includes a dynamic and a static model. The dynamic model is a generic description of the sequential nature of SEDA. The static model, on the other hand, is a very specific description of our chosen problem domain, comparison of expected performance of queueing networks. We close by arguing that the proposed SEDA framework is a good one based on its properties.

To sum up, the organization of this paper is to first define the basic terms in SEDA in Section 3, followed by a framework consisting of the dynamic and static models of the SEDA task in Section 4. Then Section 5 argues that this is a good framework.

3 DEFINITION OF TERMS

Our world of SEDA consists of six *components* which form the basis of the SEDA framework. They are:

1. **SYSTEM:** a black box with one or more parameters, which takes prescribed input and produces corresponding output.

A system in our world is a collective term, such as the M/M/1 system, which represents a class of similar system instances with different parameter values. The term system as it is commonly used refers to one instance of a class of systems in our SEDA world.

2. **PARAMETER:** a collection of constants that define an instance of a system.

The instances of one system are distinguished by a common set of parameters, but with different values.

3. **RESOURCE:** a constrained quantity that is necessary to solve a problem. The resources considered in our SEDA world are (real) time, the computer system, and the user.

We believe that real time is the most important resource in SEDA and other resources, such as CPU time (power), can be expressed in terms of it. The computer system and the user contribute information, including knowledge and decisions, to this SEDA problem-solving environment.

4. **DESIGN:** the design in our SEDA world consists of the number of replications, the stopping time for each replication, the random number assignment, and the data aggregation.

Data aggregation is any reduction of the raw output data which may be needed if we are not be able to efficiently keep and utilize all the raw data or to transform the data into a more useful form. Within our world, data aggregation includes batch size and data deletion, for batching and weighting the data, respectively. Other types of aggregation could be included.

5. **DATA:** all of the simulation output.

The data are characterized by a multivariate joint distribution that is typically unknown to us.

6. **ANALYSIS:** deriving statements about systems. The term "statement" will be formally defined in Section 4.1.

The above definitions for the components in our SEDA world establish common ground for further discussion. Although they may not be exhaustive, these components represent our view of the SEDA world. These components are strong organizing principles for SEDA. Their definitions are precise, but not at a working level. Using these organizing principles, the framework of SEDA is presented next.

4 A FRAMEWORK FOR SEDA

We now present a framework for the SEDA task consisting of a *dynamic* and a *static* model. The dynamic model in Section 4.1 defines fundamental SEDA concepts, and describes how SEDA proceeds dynamically through the primatives consisting of the concepts and the components in our SEDA world. The static model in Section 4.2 presents the structure of simulation problems, SEDA building blocks, and statistical procedures in a hierarchical fashion.

4.1 Dynamic Model of the SEDA Task

By the dynamic model of the SEDA task we mean a description of the interplay between the SEDA primatives in an interactive environment. The "primatives" include the SEDA components (defined in Section 3), and the concepts, introduced next. In brief, *components* are generic building blocks of SEDA while *concepts* define SEDA dynamics. The relationship between primatives is given in Section 4.1.1 with an illustrative example in Section 4.1.2.

4.1.1 Fundamental SEDA Concepts

The SEDA *concepts* are:

1. **SYSTEM INSTANCE:** a system with a set of fixed values of the system-dependent parameters.

The difference between a system and a system instance is that several system instances could be de-

rived from the same system with different values of the parameters.

2. **STATEMENT:** any declaration about the system instances.

The sources of statements are prior knowledge and experimental analysis. Prior knowledge includes knowledge of the problem domain and of similar classes of systems, while experimental analysis draws intermediate and final conclusions based on data.

3. **SCOPE:** the subset of data upon which a statement is based.

The scope of a statement therefore implies the applicable system instances.

4. **PROCEDURE:** a function of data and statements that produces a new statement.

Parametric and non-parametric statistical procedures are two major categories of procedures.

5. **EXPERIMENT:** executing a system instance according to a design to produce data.

The terminology used for these concepts may not be standard, but their meanings (definitions) are unambiguous in our SEDA world. As opposed to the SEDA components defined in Section 3, the definitions of these concepts are at a working level that describes the SEDA task.

Concepts and components are both SEDA primatives. The reason we separate them is because a component is more concrete and traditionally known in SEDA, while a concept is more ambiguous, confusing, and is often ignored.

The main reason why these fundamental concepts are valuable is because through defining them the components of SEDA—the system, the resource, the design, the parameter, the data, and the analysis—can be connected and reasoned in a dynamic, sequential, problem-solving process, as described in Section 4.1.2.

4.1.2 Dynamics of SEDA

Because the SEDA problem solving is basically an iterative process, we first describe the generic cycle. Then, based on this generic cycle, the sequential nature of the SEDA task is presented.

One SEDA-cycle describes the possible activities and interactions between the primatives. In brief: Resources + Pre-Analysis + Experiment Design + Statements + Data + Post-Analysis $\Longrightarrow$ New statements + Scope.

Stated differently, the SEDA-cycle proceeds as follows: under the real time constraint, a pre-analysis is performed for deriving new statements and an experiment design that generates the data. Then, a post-analysis is performed using both the data and available statements for producing new statements and their scope. The computer system and the user provide the necessary knowledge and decisions during this SEDA-cycle within their capabilities.

Although a generic SEDA-cycle attempts to describe all the possible interactions between these primatives, all the activities will not necessarily happen in every SEDA-cycle. Nevertheless, one or more statements are produced in every SEDA-cycle.

Statements are produced sequentially during the iterative SEDA-cycles, so their scope may reach backward several cycles. In other words, any statement may depend on previous statements. For example

$$statement_5 = proc_5(data_5, proc_4(data_4 ...))).$$

Notice that the system instances and the subset of data are embedded within this sequential representation of the SEDA task.

Theoretically, if there is no resource limit, the true values of the system performance measures of interest can be obtained. In reality this sequential SEDA process has to stop before the available real time runs out. Accordingly, the goal of SEDA problem solving, in practice, is to produce a simulation result within a desired error level under the real time constraint.

A final comment is that these fundamental concepts and components are well-defined and precise elements of the dynamic SEDA task. However, in real SEDA problems, distinct concepts or components may not be easily identified and separated. For example, experiment design and analysis are tightly bound together and thus it is not easy to clearly identify which is which. Also, procedures very often affect the experiment design and thus they do not necessary follow the process exactly. An expert understands better than a novice about this tight coupling, which is the ability to look ahead.

Yet this generic SEDA cycle also demonstrates the strength of our SEDA primatives which represent the complex SEDA process in a simple but well-defined manner.

4.1.3 An Example

The sequential nature of SEDA can be illustrated by a few segments of a simplified example, which is based on a protocol script from a real simulation problem solved by an SEDA expert.

A user wants an SEDA expert to help find the smallest average waiting time among three queueing system instances with mean interarrival times and mean service times (1.05, .9), (1.0, .8), and (.9 , .7), respectively. Since the final report will be due within

eight hours, the user can only spend four hours of simulation study before writing it.

System: the queueing system.

System Instances: the queueing system with three different sets of values for the system parameters, interarrival time and mean service time.

Parameters: (interarrival time, mean service time) = (1.05, .9), (1.0, .8), and (.9, .7) for the three system instances, respectively.

Resource: four hours of real time and the user and the SEDA expert, where the user may supply the knowledge about the system and major SEDA decisions, and the SEDA expert may supply the knowledge of SEDA and queueing systems.

"I (the SEDA expert) am going to make replications and look at the bias for one of the system instances. I have no other reasons for choosing this approach but to get a better look at the bias.... Look at the traffic intensity and find out which is the most congested system. I will use system instance 1 since it has the highest traffic intensity."

Analysis: the quote above is part of the pre-analysis.

Resource: the SEDA expert who is contributing his knowledge of SEDA and queueing systems.

System Instance: the queueing system with the first set of parameters.

"Let me make a quick run of 2 replications of 2,500 observations for system instance 1.... It did not take long (about 3 seconds) to make 2 replications."

Design: the number of replications (2) and the stopping time for each replication (2,500 waiting times).

Data: the simulation output as planned in the above design.

Experiment: Executing system instance 1 with the design of 2 replications each with 2,500 waiting times.

Statement: "3 seconds for 2 replications of 2,500 observations for system instance 1."

Scope: system instance 1 with the data in the statement.

"Look at the data. Some bias at the beginning.... It seems to climb quickly. I am taking a little gamble to do 10 replications each with 200 observations."

Procedure: visual inspection of the trend of data.

Statement: "Some bias at the beginning." It has the same scope as the previous statement.

Analysis: inspecting the trend of the data and producing a statement about a new design.

Design: the new number of replications (20) and stopping time (200 observations).

The following statements with their implicit scope illustrate the sequential nature of the SEDA expert's problem-solving process:

Statement 4 "The three queueing system instances are logically similar."

Statement 20 "I am sure that the appropriate initial bias deletion point is 500 for system instance 1."

Statement 21 "I spent too much time in determining the initial bias deletion for system instance 1 and I have very limited time available now."

Statement 22 "I will use this deletion point for all three queueing system instances."

Statement 23 "The current data, which looks exponential and has lag-1 correlation 0.51, fail the normal and the independence assumptions for system instance 1."

One major element that distinguishes an expert from an intermediate user is that the expert is able to manage the resource limit, real time, in simulation problem solving, but the intermediate user usually is not. Also, the expert reasons by combining statements to obtain a desirable new statement. Those experts who well understand the sequential nature of simulation problem solving know how to achieve the goal effectively and efficiently with the available resources.

4.2 Static Model of SEDA

The static model is a description of the SEDA structure for solving a particular class of problems, in our case the problem of comparing expected performance across queueing-network models. Our static structure is represented as a three-layer model.

To help simulation users, we need a static model that is not only based on a solid set of primatives, but is also easily understood and learned by simulation users. The static model need not be unique, but

it should be able to be explained by the SEDA primatives, and be able to explain the actual design of an SEDA-CS.

Accordingly, the difference between the dynamic model and the static model, in addition to the dynamic verses static view, is that the dynamic model forms a high-level abstraction for the nature of SEDA that will not change over time, while the static three-layer model is a lower-level representation of the SEDA task that may change as the technologies or methodologies evolve.

Because this static model is a more detailed description, we present it within our research focus, queueing-network simulation. Our static model of the SEDA task consists of a classification of simulation problems, a decomposition of simulation tasks, and a hierarchy of simulation procedures (a three-layer breakdown of SEDA is presented in Figures 1–3).

Layer one - classification of simulation problems: Figure 1 implies that any given simulation problem on the top node can be classified into a desired solution on the bottom nodes. Within the classification scheme, a comparison with known alternatives problem can be divided into three subproblems: basis of comparison, design, and analysis.

Before the basis of comparison, we first classify a simulation problem into either a comparison problem with known alternatives, or a comparison problem without a known alternative. In our view, all simulation problems are comparison problems. The main difference is to what a system instance is compared. If a system instance is only evaluated to compare to the true value of its performance measure of interest, which is unknown but fixed, then this is a comparison without a known alternative. On the other hand, when a system instance is compared to other known alternatives, then this is a comparison with known alternatives.

Layer two - decomposition of simulation tasks: Since the representation of the task attempts to consider the subtasks and issues involved at different stages of the simulation problem solving, and since design and analysis are two iterative processes, a lower level-representation of the subtasks for design and analysis can be represented as in Figure 2.

Notice that in Figure 2 the comparison problem with a known alternative is further divided into two independent modules: the initial-bias recognition problem and the core comparison problem. Under these two subproblems are, then, the subtasks of design and analysis. One important point is that the subtasks under design and analysis can be derived from the primatives as defined earlier. This is what we think is important in designing a CS: A fundamental conceptual understanding of a complex activity for explaining the empirical model based on some task analysis.

Layer three - hierarchy of simulation procedures: The third layer is the procedure tree in Figure 3. Figure 3 first classifies a simulation problem into either an output-analysis problem or an initial-bias recognition problem. The decomposition matches Figure 1 until the "means-comparison problem." From here on, a statistical procedure can be determined by branching down to the bottom levels of this hierarchy.

The procedures in Figure 3 are just one set of basic procedures used in comparison with known alternatives. There are many other procedures used by other experts, which are expert-dependent, and are not shown here. Accordingly, although this layer is necessary for actually solving a problem, the procedures are not exhaustive. Thus, it should be emphasized that the methodologies are evolving and they do not encompass all the details an expert considers in the problem-solving process.

The advantage of this layer is that it can easily map to the classification of simulation problems in Figure 1. Therefore, our static model can encompass a large ranges of statistical procedures and analysis.

Figures 1–3 are not complete. One reason is that we only focus on a limited problem, that is, the means-comparison problem with known alternatives. Another reason is that we provide only one possible scheme to represent the fundamentals of simulation problems. However, this scheme is not unique. Moreover, even within this scheme our colleagues may fill in or replace some of the details as the methodology and research advance. In other words, we are more concerned with the fundamental structure and the sketch of SEDA rather than the details within the structure. Consequently, the SEDA primatives should embrace any new methodology and not be limited to the procedures provided in the third layer of the SEDA model.

5 JUSTIFICATION

A prototype SEDA-CS is being developed based on our framework, and an evaluation will be performed to demonstrate the feasibility and appropriateness of this SEDA framework. We briefly argue here that the proposed SEDA framework is a good one.

We believe that an SEDA framework should be self-contained, simple, specific only when necessary, and comprehensive.

1. **Self-contained:** Our SEDA framework has in total eleven primatives that are briefly but precisely

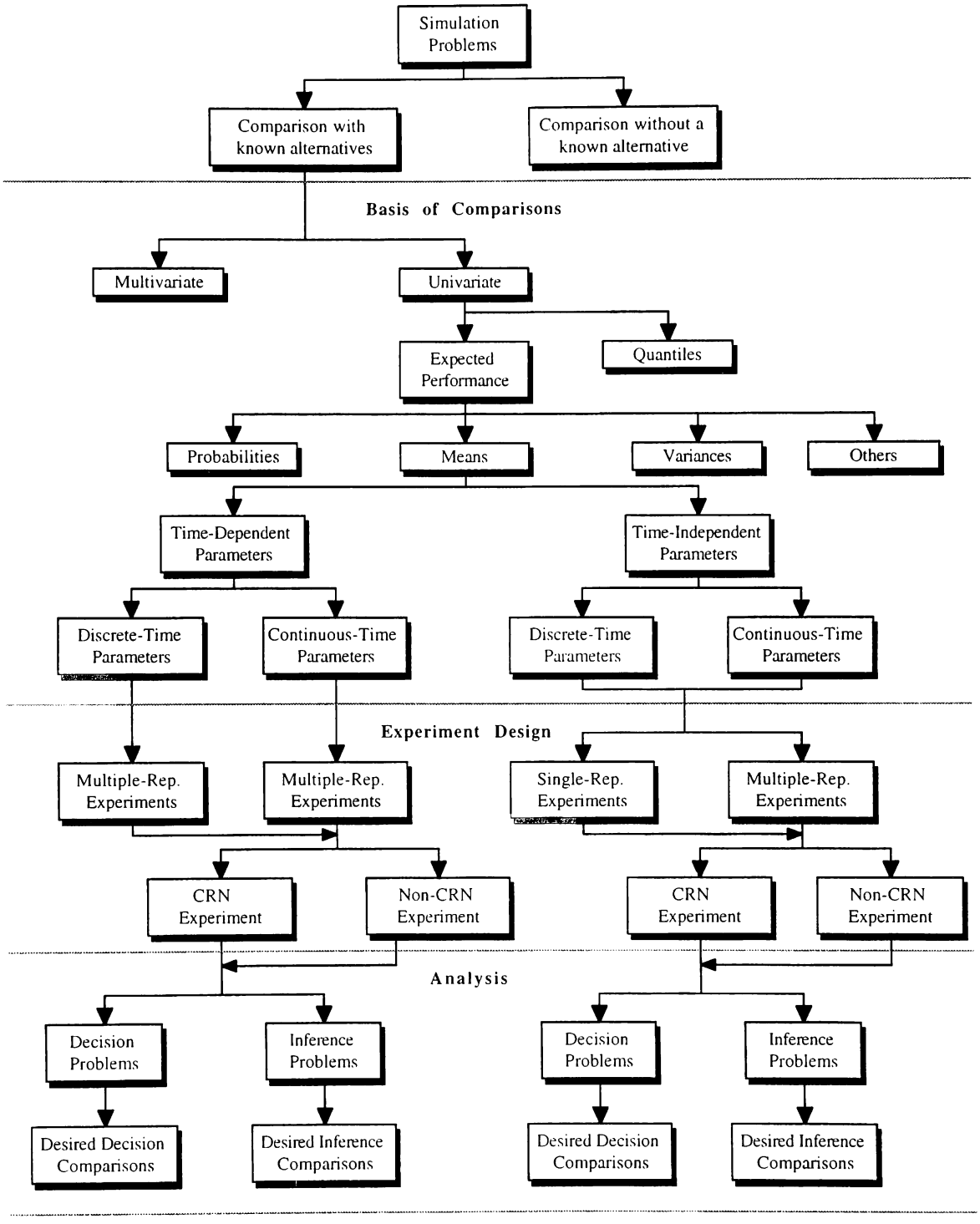

Figure 1: A Classification of Simulation Problems

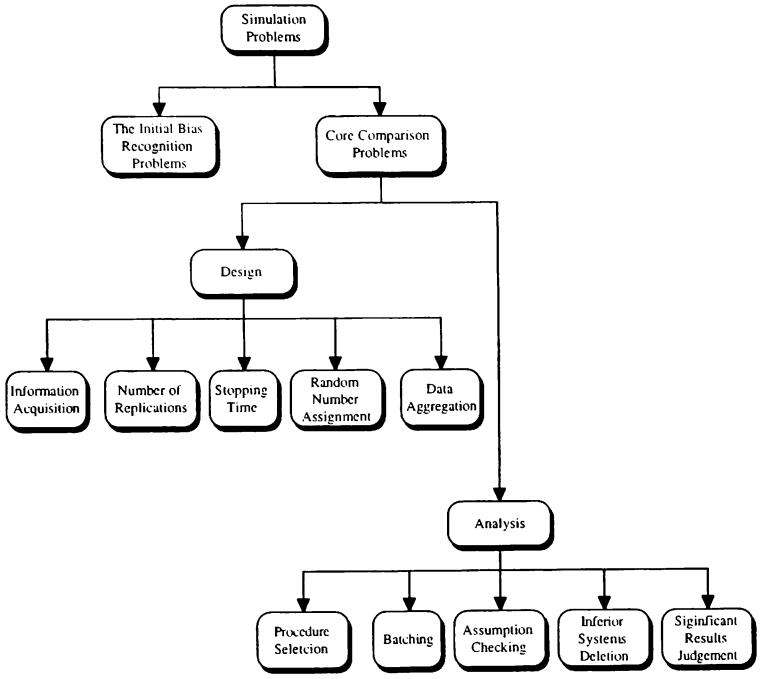

Figure 2: A Decomposition of Simulation Problems

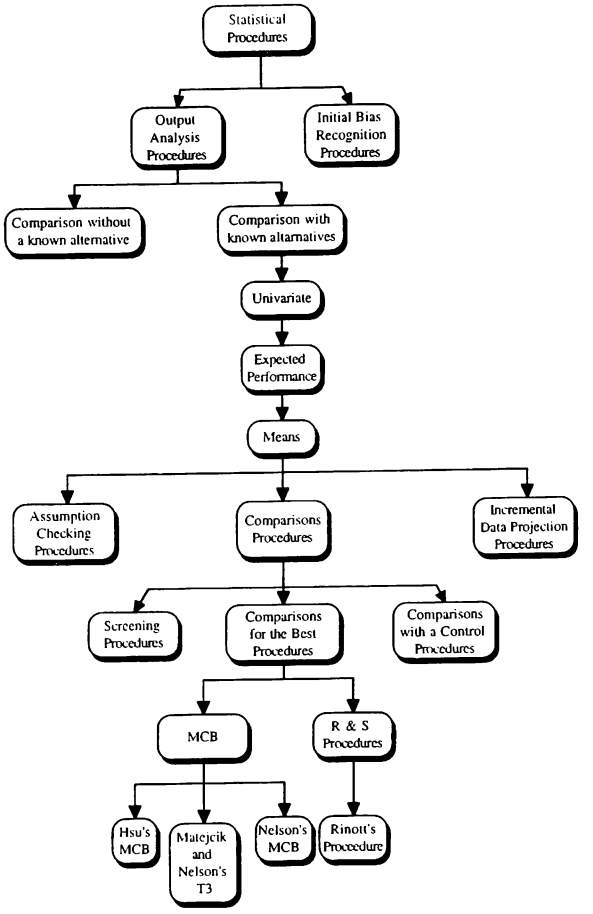

Figure 3: A Hierarchy of Simulation Procedures

defined. These primatives help to derive and develop both the dynamic model and the static model. Since our framework is developed based on our SEDA world, it is self-contained.

2. **Simple:** Our SEDA framework is simple since it contains only the static and dynamic model of the SEDA task. Each of them has only one central theme: sequential natural and three-layer structure, respectively.

3. **Specific Only When Necessary:** Our SEDA framework involves only fundamental elements without any implementation detail. Moreover, all these fundamental elements are required to describe the SEDA framework. Although there are some specific details in our SEDA framework, they are necessary to completely describe the static model in the bottom level of the hierarchy of simulation statistical procedures.

We believe that the dynamic model will not be affected by technologies and methodologies over time, and only the bottom level of the static model might be

4. **Comprehensive:** Our framework is comprehensive in that it is compatible with any statistical procedure. It covers everything from a complete formal one-step data analysis to sequential data analysis.

A good representation not only needs to have the characteristics defined above, but also to be arranged in a smooth sequence for easy comprehension. To this end, the dynamic and static models are decomposed in a hierarchical fashion. Then, the bottom-up approach is used to introduce the dynamic model starting from the SEDA concepts, and then the generic SEDA-cycle and sequential statements of iterative SEDA-cycles. On the other hand, a top-down approach is used to describe the three-layer structure from the top layer to the bottom layer.

REFERENCES

Dyer, J., J. C. Bean, L. S. Dewald, A. H. Gepfert, and A. Odoni. 1993. Suggestions for an MS/OR Master's degree curriculum. *OR/MS Today* 2:22–31.

Freeman, P. 1980. The nature of design. In *Tutorial on Software Design Techniques*, ed. P. Freeman and A. Wasserman, 46–53. IEEE.

Nelson, F. N., D. A. Bradbard, W. N. Ledbetter and J. F. Cox. 1987. Use of operations research in production management. *Prod. Invent. Mgmt.* 28:59–62.

O'Keefe, R. M. 1986. Simulation and expert systems–A taxonomy and some examples. *Simulation* 46:10–16.

Shannon, R. E. 1985. Intelligent simulation environments. In *Proceedings of Intelligent Simulation Environments*, ed. P. A. Luker and H. H. Adelsberger, 150–156. SCS, San Diego, California.

Taylor, R. and R. D. Hurrion. 1988. An expert advisor for simulation experimental design and analysis. In *AI and Simulation*, ed. T. Henson, 238–244. SCS.

Thomas, G. and J. A. DaCosta. 1979. A sample survey of corporate operations research. *Interfaces* 9:104.

AUTHOR BIOGRAPHIES

YU-HUI TAO is a Ph.D. candidate in the department of Industrial, Welding and Systems Engineering at the Ohio State University. He received his M.S. from the Ohio State University in 1989. Currently, he works as a software consultant in a bank for developing a rule-based computer system for detecting credit-card frauds. He is interested in system design and software development applying methodologies in Operations Research, Cognitive Engineering, and Computer Science.

BARRY L. NELSON is an Associate Professor in the Department of Industrial, Welding and Systems Engineering at The Ohio State University. His research interests are experiment design and analysis of stochastic simulations. He is Past-President of the TIMS College on Simulation, Simulation Area Editor for *Operations Research* and Associate Editor for *ACM TOMACS*.