IIE Transactions
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713772245

Computer-assisted simulation analysis

Yu-Hui Tao[a]; Barry L. Nelson[b]
[a] Computer People Consulting Services, Westerville, OH, USA [b] Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL, USA

To cite this Article Tao, Yu-Hui and Nelson, Barry L.(1997) 'Computer-assisted simulation analysis', IIE Transactions, 29: 3, 221 — 231
To link to this Article: DOI: 10.1080/07408179708966329
URL: http://dx.doi.org/10.1080/07408179708966329

IIE Transactions (1997) **29**, 221–231

Computer-assisted simulation analysis

YU-HUI TAO[1] and BARRY L. NELSON[2]

[1]*Computer People Consulting Services, 1257 Wallasey Drive, Westerville, OH 43081, USA*
[2]*Department of Industrial Engineering and Management Sciences, Northwestern University, 2225 North Campus Drive, Evanston, IL 60208, USA*

Received August 1995 and accepted August 1996

We propose a framework for understanding the complex activities of Simulation Experiment Design and Analysis (SEDA), a framework that provides a basis for developing SEDA software. This paper presents the framework and illustrates it with a brief example. The definitions of six SEDA components form the core of the framework. The framework itself includes a dynamic model and a static model. The dynamic model is a generic description of the sequential nature of SEDA. The static model, in contrast is a very specific description of the SEDA computer system structure for our chosen problem domain: comparison of queueing networks in terms of their expected performance. We argue that the proposed SEDA framework is a good one on the basis of its properties and by comparing it with related frameworks. We close by briefly summarizing our prototype SEDA software, which was used to evaluate our framework.

1. Introduction

Our goal is to help simulation users with Simulation Experiment Design and Analysis (SEDA) through the assistance of a computer system (CS). However, instead of simply developing a SEDA-CS, our primary interest is in understanding, constructing and validating a high-level framework for the complex SEDA problem-solving process. We then demonstrate the efficiency of our framework through a prototype implementation. Our premise is that by starting with a framework for SEDA, we make it easier for developers and end-users to work together to produce even better implementations of such SEDA-CSs.

Software is already available that provides a simulation problem-solving environment. Intelligent front ends [1, 2] and program generators [3–7] are useful tools for interactive simulation modeling with existing simulation software, and integrated simulation systems [8, 9] provide artificial intelligence-assisted simulation environments with consistent specification and internal representation for both simulation modeling and analysis. Most of the available simulation analysis tools help simulation users either by automating the process or by focusing on a specific field, such as manufacturing [10–13].

Our focus is on the statistical issues in simulation problem solving. Although standard statistical analysis packages provide a rich set of statistical tools for their users, the user has to know both statistics and issues specific to SEDA to perform simulation problem solving. Our SEDA framework addresses this interaction.

Three papers describe work that is closely related to ours. Mellichamp and Park [14] organized statistical procedures according to simulation problems, and developed a 'Statistical Expert System for Simulation Analysis' to assist simulation users in selecting the appropriate procedures for each subproblem of the targeted SEDA problem. Taylor and Hurrion [15] justified the use of an expert system framework for simulation experimentation, and developed the 'Warwick Expert Simulation' to assist simulation users in SEDA problem solving with embedded problem-solving expertise. Ramachandran *et al.* [16] described a framework for an expert postprocessor for simulation output analysis, with emphasis on interactive model validation, to be used in their 'Intelligent Simulation Generator'.

The literature supports the idea that a CS can help simulation users in all aspects of simulation problem solving. In particular, the three papers mentioned above demonstrate the feasibility of such interactive computer-assisted SEDA applications. However, the fundamental characteristics of the SEDA task are not clearly or completely identified in these papers, and thus we believe that there is room for substantial gains by defining these fundametal characteristics. We define the basic components of SEDA in Section 2, followed by a framework consisting of a dynamic model and a static model of the SEDA task in Section 3. Then Section 4 argues that this is a good framework by comparing it with related CSs. In Section 5 we briefly describe our prototype implementation of an SEDA-CS; our conclusions are in Section 6. Parts of this paper are based on Tao and Nelson [17].

2. SEDA Components

The SEDA environment consists of six *components* that form the basis for the SEDA framework. They are:

1. **System:** a black box with one or more parameters, which takes prescribed input and produces corresponding output. In our definition, a system 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 definition. For example, we may be comparing three *M*/*M*/1 instances, each with its own values of mean interarrival time and mean service time. Because these three instances all have the same basic structure, except for different paired values of mean interarrival time and mean service time, the three *M*/*M*/1 instances belong to the same class of systems. Of course, our definition of system allows for much more complex models and parameters, including networks of queues for which the parameters might be numbers of servers, queue disciplines or even number of queues.
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. The constants need not have a physical meaning; e.g., a parameter could be the queue discipline, with '1' corresponding to first-come-first-served.
3. **Resource:** a constrained quantity that is necessary to solve a problem. The resources in SEDA are real time, the computer system and the user. We believe that real time is an important resource in SEDA; other resources, such as CPU time, 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 consists of the number of replications, the stopping time for each replication, the random number assignment and the data aggregation technique. Data aggregation is any reduction of the raw output data that may be needed if we are not able to efficiently keep and utilize all of it, or any transformation of the data into a more useful form. Data aggregation includes batch size and data deletion, for batching and weighting the data, respectively. Other types of aggregation could also 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 3.1.1.

The definitions of the SEDA components establish common ground for further discussion in this paper. They represent our view of SEDA. Their definitions are precise, but they are not useful at a working level. However, on the basis of these abstract components, the SEDA framework can be derived.

3. A framework for SEDA

We now present a framework for the SEDA task consisting of a *dynamic* model in Section 3.1 and a *static* model in Section 3.2. The dynamic model defines how SEDA problem solving proceeds, whereas the static model provides the infrastructure to facilitate attacking an SEDA problem. In other words, the static model provides the system of roads, and the dynamic model provides the transportation, toward the destination of solving an SEDA problem. Before introducing the dynamic model, Section 3.1.1 defines five fundamental SEDA *constructs* that are based on the six components in Section 2; these constructs describe how SEDA proceeds dynamically.

3.1. *Dynamic model of SEDA*

By a dynamic model of the SEDA task we mean a description of the interplay between the SEDA primitives in an interactive environment. The 'primitives' include the SEDA components (system, parameter, resource, design, data and analysis as defined in Section 2) and the constructs (system instance, statement, scope, procedure and experiment as defined below). In brief, components are generic building blocks of SEDA dynamics, whereas constructs define the movement of SEDA dynamics. The relationship between primitives is illustrated by an example at the end of this subsection.

3.1.1. *Fundamental SEDA constructs*

The SEDA constructs are:

1. **System instance:** a system with a set of fixed values for the system-dependent parameters. The difference between a system and a system instance is that several system instances could be derived from the same system with different values of the parameters. In other words, system instance is derived from both the system and the parameter components.
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, whereas experimental analysis draws intermediate and final conclusions based on data.
3. **Scope:** the subset of data on 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, but the definition is broad enough to include the visual inspection of a plot.

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

The terminology used for these constructs may not be entirely standard, but their definitions are unambiguous in this paper. As opposed to the SEDA components defined in Section 2, the definitions of these constructs are at a working level that describes the SEDA task. The reason we separate them is because a component is more concrete and traditionally known in SEDA, whereas a construct is more ambiguous, confusing and is often ignored. Constructs and components are both SEDA primitives of equal importance. The constructs are valuable because, through defining them, the components of SEDA – the system, the resource, the design, the parameter, the data, and the analysis – can be connected in a dynamic, sequential, problem-solving process, as described below.

3.1.2. *Dynamics of SEDA*

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

One SEDA-cycle describes the possible activities and interactions between the primitives. In brief, it is as shown in Fig. 1.

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

Although a generic SEDA-cycle attempts to describe all the possible interactions between these primitives, it is not necessary that all the activities happen in every SEDA-cycle. Nevertheless, one or more statements must be produced in every SEDA-cycle, which is what keeps the SEDA process moving forward. Statements are produced sequentially during the iterative SEDA-cycles, so their scope may reach backwards several cycles. In other words, any statement may depend on previous statements. For example, represented in a functional notation,

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

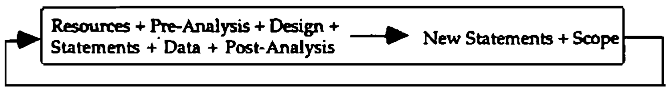

Resources + Pre-Analysis + Design + Statements + Data + Post-Analysis
New Statements + Scope

Fig. 1. SEDA-Cycle Unit.

Notice that the system instances and the data are embedded within this sequential representation of the SEDA task. The sequential nature of experimentation is an important aspect of the dynamics of SEDA. However, our definition of 'sequential' is broader than the classical statistical definition, in that we allow backtracking and interaction, which means that the analysis path is not necessarily fixed.

Theoretically, if there is no resource limit, then the true values of the system performance measures of interest can be obtained. In reality this sequential SEDA process has to stop when 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.

Although these fundamental constructs and components are well-defined and precise elements of the dynamic SEDA task, in real SEDA problems distinct constructs or components may not be easily identified or separated. For example, experiment design and analysis are often tightly coupled, and thus it is not easy to clearly identify which is which, or to match any possible design with any possible analysis. Also, the procedures that are chosen will very often affect the experiment design. The skill to look ahead to manage this tight coupling during SEDA problem solving is typically lacking in a novice. We argue below that nearly all simulation experimentation, no matter how complex, can be expressed in this simple but well-defined manner, which therefore provides an ideal structure on which to build a CS that will aid the novice as well as the sophisticated user.

3.1.3. *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 simulation problem solved by an SEDA expert. This example is a real SEDA problem-solving session that we conducted to explore SEDA activities. The protocol script illustrates the relevant thought process that the user/expert would undergo. Each quotation represents the expert thinking out loud as part of the SEDA process. The problem is as follows: A user wants an SEDA expert to help compare the expected waiting times between three queueing system instances that are identified by their mean interarrival times and mean service times: (1.05, 0.9), (1.0, 0.8), and (0.9, 0.7), respectively. Because the final report will be due within 8 hours, the user can only spend 4 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: mean interarrival time and mean service time.
Parameters: (mean interarrival time, mean service time) = (1.05, 0.9), (1.0, 0.8) and (0.9, 0.7) for the three system instances, respectively.

Resource: 4 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 quotation 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 two replications of 2500 observations for system instance 1.... It did not take long (about 3 seconds) to make two replications.'

Design: the number of replications (2) and the stopping time for each replication (2500 waiting times).
Data: the simulation output as planned in the above design.
Experiment: Executing system instance 1 with the design of two replications each with 2500 waiting times.
Statement: 3 seconds for two replications of 2500 observations for system instance 1.
Scope: system instance 1 with the data in the statement.

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

Procedure: visual inspection of the data.
Statement: 'Some bias at the beginning.' This statement 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 implied 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 will use this deletion point for all three queueing system instances.'
Statement 22: "The current data, which look exponential and have lag-1 correlation 0.51, fail the normal and the independence assumptions for system instance 1.'

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 and within the available real time.

3.2. *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. Choosing a problem domain was necessary so that we could incorporate domain knowledge and limit the number of statistical procedures in our prototype CS. However, these are not limitations of the framework itself.

Our static structure is represented as a three-layer model. The static model need not be unique, but it should be able to be explained by the SEDA primitives, and be able to explain the actual design of an SEDA-CS.

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

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. Figs 2–4 represent this three-layer breakdown. Some features that are outside the scope of this paper are omitted from the figures because of page-size limitations.

Layer one: classification of simulation problems. Fig. 2 implies that any given simulation problem on the top node can be classified into a desired type of solution on the bottom nodes. Within the classification scheme, a comparison-with-known-alternatives problem can be divided into three subproblems: basis of comparison, experiment 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 between simulation problems is with what a system instance is compared. If a system instance is evaluated only to estimate the true value of its performance measure of interest, which is unknown but fixed, then this is a comparison without a known alternative. However, when a system instance is compared with other known alternative system instances, then this is a comparison with known alternatives.

Layer two: decomposition of simulation tasks. The SEDA task can be broken down into subtasks and issues that are relevant at different stages of simulation problem solving. Design and analysis are critical high-level SEDA

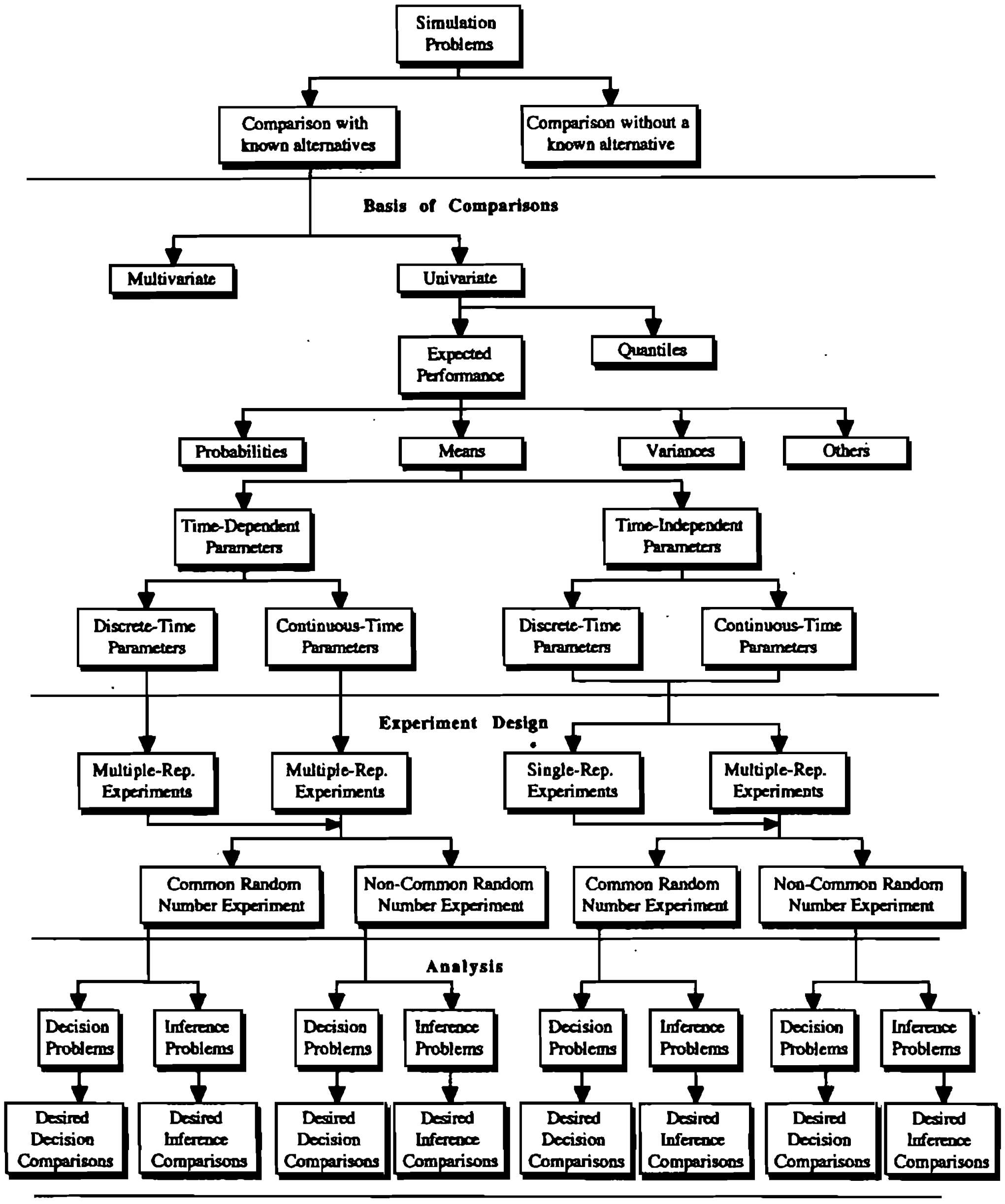

Simulation Problems
Comparison with known alternatives
Comparison without a known alternative
Basis of Comparisons
Multivariate
Univariate
Expected Performance
Quantiles
Probabilities
Means
Variances
Others
Time-Dependent Parameters
Time-Independent Parameters
Discrete-Time Parameters
Continuous-Time Parameters
Discrete-Time Parameters
Continuous-Time Parameters
Experiment Design
Multiple-Rep. Experiments
Multiple-Rep. Experiments
Single-Rep. Experiments
Multiple-Rep. Experiments
Common Random Number Experiment
Non-Common Random Number Experiment
Common Random Number Experiment
Non-Common Random Number Experiment
Analysis
Decision Problems
Inference Problems
Desired Decision Comparisons
Desired Inference Comparisons

Fig. 2. Classification of simulation problems.

tasks; a lower-level representation of the subtasks for design and analysis can be represented as in Fig. 3.

Notice that in Fig. 3 the comparison problem with a known alternative is further divided into two independent modules: the initial-bias recognition problem and the core comparison problem (note that we do not intend to imply that all simulation problems are divided into these two categories). Under these two subproblems, then, are the subtasks of design and analysis. One important point is that the subtasks under design and analysis can be derived from the primitives 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 Fig. 4. Fig. 4 first classifies a simulation problem into either an output-analysis problem or an initial-bias recognition problem. The decomposition matches Fig. 2 until the 'means-comparison problem'. From there on, a statistical procedure can be determined by branching down to the bottom level of this hierarchy.

The procedures in Fig. 4 are just one set of basic procedures used for comparison with known alternatives. There are many other procedures used by other experts. Accordingly, although this layer is necessary for actually solving a problem, the procedures are not exhaustive or fixed. The reader can ignore any unfamiliar terminology in Fig. 4 because it is for illustration only.

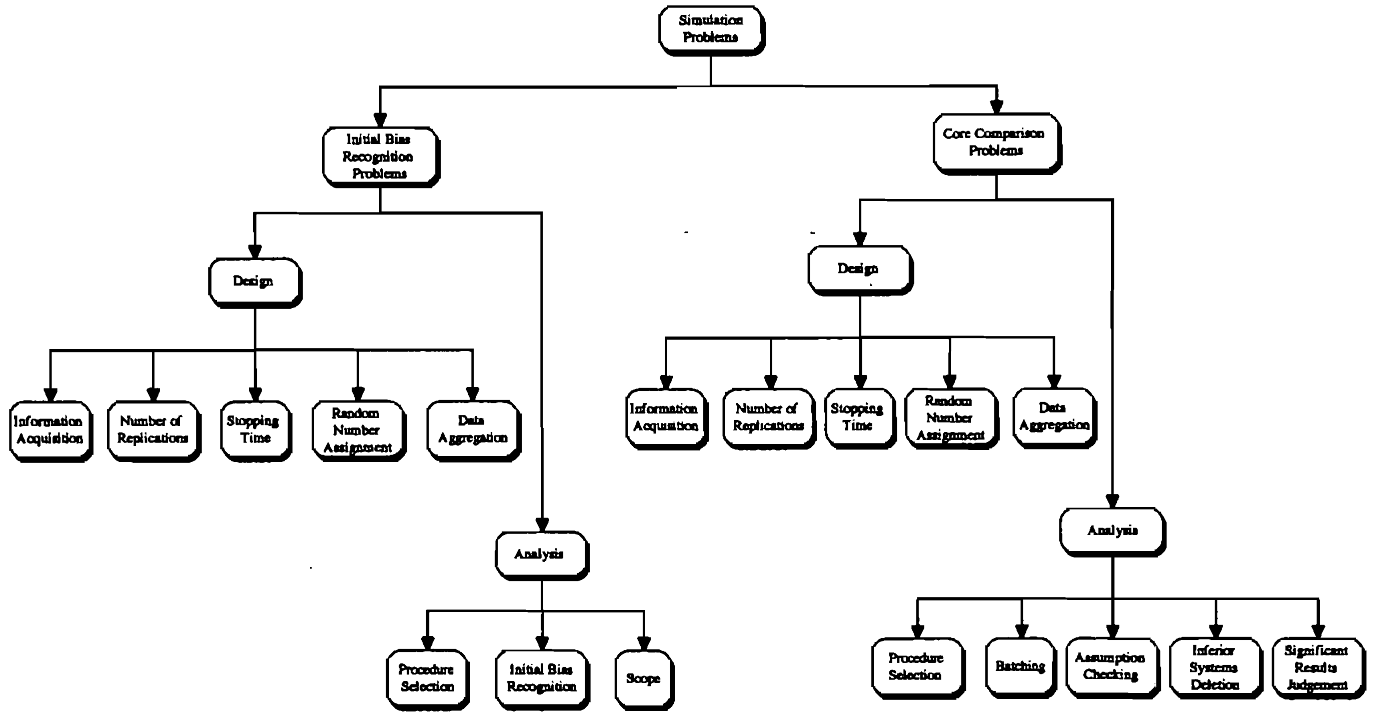


Fig. 3. Decomposition of simulation tasks.

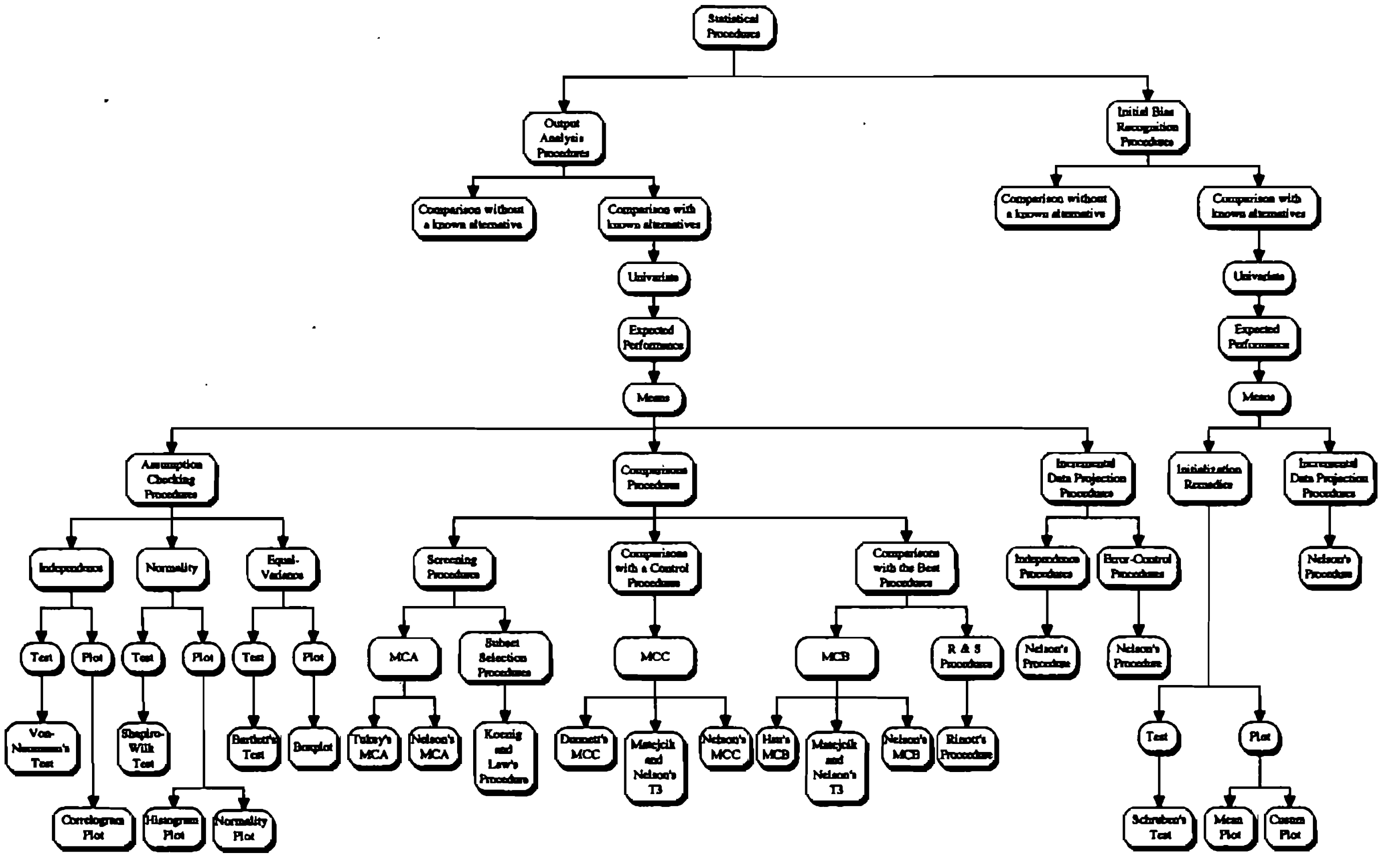


Fig. 4. Hierarchy of simulation procedures.

The value of this third layer is that the procedures are clustered based on the classification of simulation problems in Fig. 2. For example, both output-analysis and initial-bias-recognition nodes branch down to the 'means' procedures as in Fig. 2. Therefore our static model can encompass a large range of statistical procedures and analysis.

Figs 2–4 are not complete. One reason for this 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. Moreover, even within this scheme our colleagues may fill in or replace some of the details as the methodology and technology advance. In other words, we are concerned more with the fundamental structure and the sketch of SEDA than with the details within the structure. Consequently the SEDA primitives should embrace any new methodology and not be limited to the procedures provided in the third layer of the SEDA model.

4. Justification

We briefly argue that the proposed SEDA framework is a good framework in Section 4.1. We examine and evaluate existing SEDA applications in terms of this framework in Section 4.2.

4.1. *Supporting arguments*

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

Self-contained: our SEDA framework has in total 11 primitives that are briefly but precisely defined. These primitives help to derive and develop both the dynamic model and the static model.

Simple: our SEDA framework is simple because it contains only the dynamic model and static model of the SEDA task. Each of them has only one central theme: sequential nature and three-layer structure, respectively.

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 describe the static model completely only at the bottom level of the hierarchy of simulation statistical procedures.

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

Comprehensive: our framework is comprehensive in that it is compatible with almost any statistical analysis. It covers everything from a formal one-step data analysis to formal sequential data analysis to an informal exploratory data analysis. Therefore, as long as distinct alternatives are simulated, this framework applies to other situations such as ANOVA or metamodeling.

For example, suppose that we were interested in knowing the relationship between the throughput rate and two factors, the queueing discipline (A) and buffer size (B), in a queueing network system and we have 2 hours of time available. A 2×2 factorial design could be used in our simulation study with two levels in each factor: FIFO (A_1) or Shortest Processing Time First (SPTF) (A_2); and large buffer (B_1) or small buffer (B_2). We could run 20 replications each of 8 hours of simulated time for each of the 2×2 cells within the 2 hours available. The finding could be that there is a significant interaction between queue discipline and buffer size.

In terms of our framework, the following constructs and components can be identified:

System: the queueing network system.
Parameter: queueing discipline with FIFO and SPTF, and buffer capacity with large size and small size.
System instance: four system instances corresponding to the 2×2 design.
Resource: 2 hours.
Design: 20 replications each with 8 hours of simulated time.
Experiment: executing the above design for 20 replications each with 8 hours of simulated time.
Data: simulation data from the experiment above.
Procedure: ANOVA.
Analysis: examining the ANOVA table and testing for significant effects.
Statement: interaction exists between queueing discipline and buffer size.
Scope: all four system instances each with 20 replications of 8 hours of simulated time.

4.2. *Related SEDA applications*

We reviewed the three related applications discussed in Section 1 in the light of our framework. All are very useful, but certain shortcomings arise because they are not based on a fundamental set of primitives.

For example, a shortcoming of Warwick Expert Simulation (WES) [15] is that it ignores the resource real time in SEDA problem solving. A consequence is that there is a chance that the SEDA will not be accomplished satisfactorily within the available time. For instance, WES could propose a 'perfect' experiment and carry out the experiment by first generating the desired amount of data without interruption for analysis. The experiment might be perfect, but the design is based on the available information up to the stage it was proposed. If WES does not know the data-generation speed and does not monitor the progress of data generation, then this experiment might exhaust all or most of the available time for further analysis or data generation. In other words, WES has no way of effectively adjusting to the time constraint during SEDA problem solving because the time resource is not part of its world view.

In addition to time management, a shortcoming of Statistical Expert System for Simulation Analysis (SESSA) [14] is that it does not account for the sequential nature of the SEDA task; it simply provides a collection of statistical procedures, which is similar to the third layer of our static model in Fig. 4. As a result, the user actually has to do most of the SEDA reasoning, which is often the most difficult part of SEDA problem solving. A potential concern when providing only a collection of statistical procedures is that methodologies are changing over time. Therefore both the system and the user may not be able to keep pace with these changes unless there is an embedded structure for organizing the simulation problems and the statistical procedures.

Expert Post-processor for Simulation Output Analysis (EPSONA) [16] is more concerned than WES and SESSA with expressing the task of simulation output analysis at a conceptual level. However, at best, EPSONA is a representation like the three-layer static model in our SEDA framework. It is not clear what the dynamics of simulation output analysis are in EPSONA and how they interact sequentially. Also, like WES and SESSA, EPSONA ignores the real-time resource.

5. Prototype SEDA-CS

To assess the usefulness of our SEDA framework, we designed, developed and evaluated a prototype CS based on it; see Tao [18] for a complete description of the CS and the experimental evaluation. Here we summarize some important aspects of our prototype. The framework presented in this paper provided the guiding principles for designing the CS, particularly in terms of conducting sequential experimentation and time management.

Our SEDA-CS targets users who have knowledge of the problem domain (in this case, queueing network simulation), simulation modeling and simulation programming, but have limited knowledge of statistics. Therefore the CS's most important function is to guide the user through one or more SEDA cycles until their comparison problem has been solved. The expertise for this SEDA-CS came from a simulation expert with knowledge of modeling, programming, and statistics, and experience with different classes of problems. The computer in use was a DECstation 3100. The software included the ULTRIX operating system, GNU C++, C/X11-based Motif graphical user interface (GUI) and some FORTRAN libraries.

Fig. 5 shows the relationships between the simulation user, the simulation models and the SEDA-CS (the detailed structure and definitions of the terminology in the figure are outside the scope of this paper; Fig. 5 is included only to provide an overview). The communication

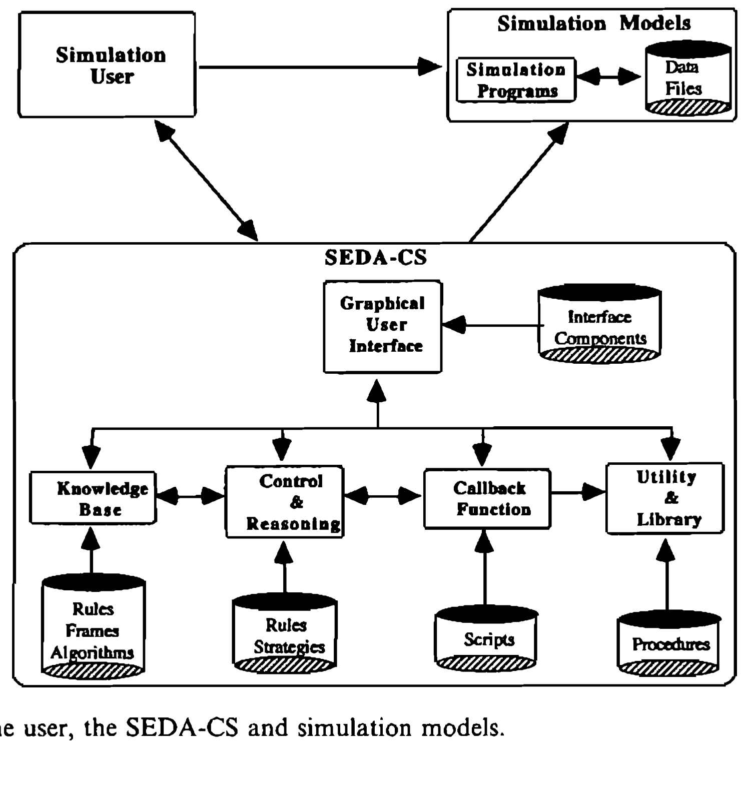


Fig. 5. Interactions between the user, the SEDA-CS and simulation models.

channels between the simulation user and the simulation models, and between the SEDA-CS and the simulation models, are one-way. However, the SEDA-CS works cooperatively with the simulation user to solve the simulation problem using their different knowledge and background. The SEDA-CS does calculations, displays results, manages data, presents alternatives with explanations, and provides suggestions and defaults when the user is unsure what to do. The user relates the results to the problem domain and makes decisions based on other external considerations.

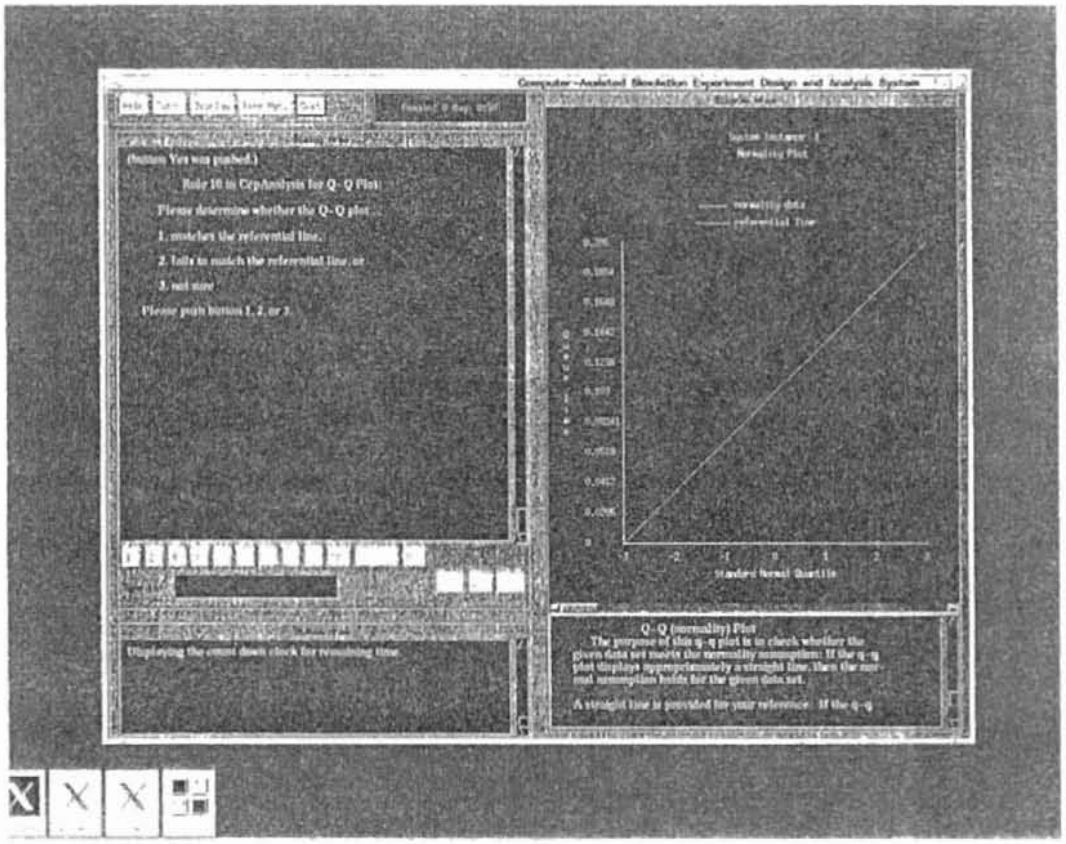
Throughout the cooperative problem-solving session there is question-and-answer dialogue between the user and the SEDA-CS. The SEDA-CS asks many questions at the beginning of the session to classify the simulation problem relative to layer one of the static model (Fig. 2). On the basis of this dialogue, the SEDA-CS produces statements about the systems and begins to break down the simulation problem into those subtasks in layer two of the static model (Fig. 3); it also proposes designs for generating data, leading the user to perform a sequence of experiments. This phase corresponds to the experiment design and analysis in layer one of the static model (Fig. 2). Before reaching a final result at the bottom node of the classification tree, many SEDA-cycles, involving the activities described in Section 3.1.2, may occur. Each cycle generates at least one new statement about the systems with corresponding scope. These statements are frequently based on the statistical procedures in layer three of the static model (Fig. 4). The statements help the user to decide what to accomplish in the next SEDA cycle, or they are the basis for recommendations by the SEDA-CS when the user is unsure what to do next.

Sequential experimentation via SEDA cycles stops when a satisfactory result is obtained or the real-time resource is exhausted. Even when time expires before a satisfactory result is obtained, the time-management features help the user to obtain as much information as possible. Time management works well in our SEDA-CS environment because it allows backtracking and interaction, meaning that the user can interrupt the experimentation and switch to a different subtask if needed.

We show two screen dumps of our SEDA-CS in Figs 5 and 6 that illustrate how our GUI works and provide examples of time management and sequential experimentation.

Fig. 6 is a screen dump of the SEDA-CS main screen. On the left-hand side of the mainscreen there are three areas:

1. **Function buttons and time display area**. These function buttons, Help, Tutor, Display, Time Mgt. and Quit are available at all times during the SEDA-CS consulting session. The time display area can display the options



(button Yes was pushed.)

Rule 10 in CrpAnalysis for Q-Q Plot:

Please determine whether the Q-Q plot ...

1. matches the referential line,

2. fails to match the referential line, or

3. not sure

Please push button 1, 2, or 3.

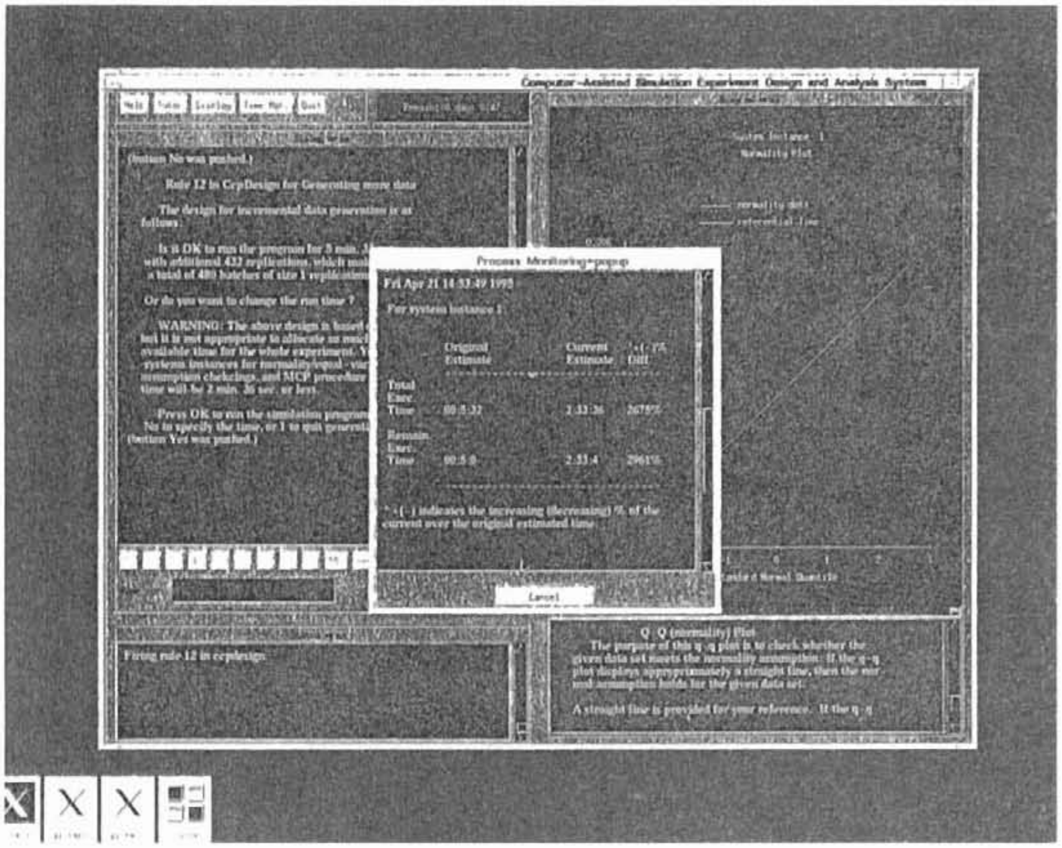
Displaying the count down clock for remaining time

Q-Q (normality) Plot

The purpose of this q-q plot is to check whether the given data set meets the normality assumption: If the q-q plot displays approximately a straight line, then the normal assumption holds for the given data set.

A straight line is provided for your reference. If the q-q

Fig. 6. SEDA-CS main screen.



```
(button No was pushed.)

      Rule 12 in CrpDesign for Generating more data

   The design for incremental data generation is as
follows:

   Is it OK to run the program for 3 min. 3
with additional 432 replications, which mak
 a total of 480 batches of size 1 replication

Or do you want to change the run time ?

   WARNING: The above design is based
but it is not appropriate to allocate so much
available time for the whole experiment. Y
systems instances for normality/equal-var
assumption checkings, and MCP procedure
time will be 2 min. 36 sec. or less.

   Press OK to run the simulation program
 No to specify the time, or I to quit generati
(button Yes was pushed.)
```

```
Fri Apr 21 14:53:49 1995

For system instance 1

          Original        Current    *+(-)%
          Estimate        Estimate   Diff
          ------------------------------------
Total
Exec.
Time      00:5:32         2:33:36    2675%

Remain.
Exec.
Time      00:5:0          2:33:4     2961%
          ------------------------------------

* +(-) indicates the increasing (decreasing) % of the
current over the original estimated time.
```

```
Firing rule 12 in crpdesign
```

```
             Q-Q (normality) Plot
   The purpose of this q-q plot is to check whether the
given data set meets the normality assumption. If the q-q
plot displays approximately a straight line, then the nor
mal assumption holds for the given data set.

A straight line is provided for your reference. If the q-q
```

Fig. 7. SEDA-CS main screen with a pop-up window.

within the Time Mgt. button, including the current clock time, the remaining time until the project is due and the project due time. In Fig. 6, 50 minutes of remaining time was displayed when this screen dump was obtained. This is one way in which the system helps the user manage the real-time resource.

2. **Dialogue area**. The upper part of the dialogue area displays instructions from the SEDA-CS to the user, whereas the bottom part of the display highlights those options available to the user. In this screen dump the SEDA-CS was asking the user to determine whether the Q-Q plot on the right-hand side of the SEDA-CS main screen matched the referential straight line. Thus the system was guiding the user towards producing a statement to complete the current SEDA cycle. A structured record of all statements produced during the session and their scope is maintained by the CS, either to aid the user in designing the experiment for the next SEDA cycle, or for use by the CS when the user appeals to it for guidance.

3. **Status area**. The status area displays the last action performed by the SEDA-CS. Viewing the remaining time was the last action taken before this screen dump.

The right-hand side of the screen is the diagram area. The upper part displays the diagram in question, whereas the lower part displays a description of the diagram. In the figure, the Q-Q (normality) plot for system instance 1 is displayed with a 45° referential line. Supplementary information about the Q-Q plot is shown below the plot.

Fig. 7 is a screen dump of the SEDA-CS main screen at the time at which data generation was performed for system instance 1. The pop-up window updates the estimated time to complete the data generation every 30 seconds, with an option to cancel data generation by pushing the Cancel button. This is the primary way that the system helps the user manage the available time: by projecting the time that will be consumed by any proposed experiment, and then updating the estimate while data generation is in progress.

In Fig. 7 the cursor is an arrow, rather than a watch, which means that the SEDA-CS was available to the user during data generation (data generation is a child process spawned from the original SEDA-CS process). The user can also revert to the UNIX operating system at any time to perform other tasks while the SEDA-CS is busy.

6. Conclusion

We have performed a formal user evaluation to understand the usefulness of our SEDA framework, which is not included in this paper; see Tao [18]. In the user evaluation we had two test cases, and six subjects each solved one of the two test cases for up to 2 hours. We explained to each subject their test case and trained them in use of the prototype SEDA-CS before the evaluation began. During the problem-solving sessions, the subjects were asked to think out loud, and every session was videotaped for transcribing verbal-protocol data that was used to analyze the results.

We were satisfied with the result of the user evaluation using our prototype SEDA-CS. However, to prove that an SEDA-CS based on our SEDA framework can help simulation users better than other related problem-solving setups, an experiment to compare users using our SEDA-CS to users using their own tools or SESSA-/WES-like systems is needed.

Acknowledgements

We thank the Departmental Editor and referee for suggestions that greatly improved the readability of the paper.

References

[1] Doukidis, G. and Paul, R. (1985) Research into expert systems to aid simulation model formation. *Journal of the Operational Research Society*, **36**, 319–325.

[2] Haddock, J. (1987) An expert system framework based on a simulation generator. *Simulation*, **48**(2), 45–53.

[3] Ford, F.R. and Schroer, B.J. (1987) An expert manufacturing simulation system. *Simulation*, **48**(5), 193–200.

[4] Haddock, J. (1988) A simulation generator for flexible manufacturing systems design and control. *IIE Transactions*, **20**(1), 22–31.

[5] Mathewson, S.C. (1984) The application of program generator software and its extensions to discrete event simulation modeling. *IIE Transactions*, **16**(1), 3–18.

[6] Paul, R.J. and Chew, S.T. (1987) Simulation modelling using an interactive simulation program generator. *Journal of the Operational Research Society*, **38**(8), 735–752.

[7] Su, H.M. and Kachitvichyanukul, V. (1989) A natural language system to aid simulation model formulation. *Simulation*, **16**(4), 535–543.

[8] McArthur, G.M., Klahr, P. and Sarain, S. (1986) ROSS: an objected-oriented language for constructing simulation, in *Expert Systems Techniques, Tools, and Applications*, Klahr, P. and Waterman, D. (eds), Addison Wesley, Reading, MA, 70–94.

[9] Sathi, N., Fox, M., Baskaran, V. and Bouer, J. (1986) Simulation craft: an artificial intelligence approach to the simulation life cycle, in *Proceedings of the 1986 Summer Computer Simulation Conference*, Society for Computer Simulation, San Diego, CA, pp. 773–778.

[10] Castillo, D. and Cochran, J.K. (1987) A comprehensive micro computer-based statistical program for analyzing simulation input and output data, in *SCS National Conference on Modeling and Simulation on Microcomputers*, Society for Computer Simulation, San Diego, CA, pp. 3–11.

[11] Hong, S., Cochran, J.K. and Mackulak, G.T. (1989) Specification of an architecture for simulation environments. in *Simulation and AI*, Webster, W. (ed.), Society for computer simulation, 77–82.

[12] Lin, C. and Cochran, J.K. (1989) An automated experimental design preprocessor for unbiased simulation analyses, in *Proceedings of the 1989 Winter Simulation Conference*, MacNair, E.A., Musselman, K.J. and Heidelberger P., (eds), IEEE, Piscataway, NJ, pp. 86–90.

[13] Thesen, A. (1989) Analysis of simulation using SANDIE, in *Proceedings of the 1989 Winter Simulation Conference*, MacNair, E.A., Musselman K.J., and Heidelberger P., (eds), IEEE, Piscataway, NJ, pp. 344–349.

[14] Mellichamp, J.M. and Park, Y.H. (1989) A statistical expert system for simulation analysis. *Simulation*, **52**(4), 134–139.

[15] Taylor, R. and Hurrion, R.D. (1988) An expert advisor for simulation experimental design and analysis, in *AI and Simulation*, Henson, T. (ed.), Society for Computer Simulation, pp 238–244.

[16] Ramachandran, V., Kimbler, D.L. and Naadimuthu, G. (1988) Expert post-processor for simulation output analysis. *Computers Industry and Engineering*, **15**(14), 98–103.

[17] Tao, Y.-H. and Nelson, B.L. (1994) A conceptual framework for simulation experiment design and analysis, in *Proceedings of the 1994 Winter Simulation Conference*,. Tew, J.D., Manivannan, S., Sadowski, D.A. and Seila, A.F. (eds), IEEE, Piscataway, NJ, pp. 681–688.

[18] Tao, Y.-H. (1995) A framework for computer assisted simulation experiment design and analysis. Ph.D. dissertation, Department of Industrial, Welding and Systems Engineering, The Ohio State University, Columbus, Ohio.

Biographies

Barry L. Nelson is an associate professor in the Department of Industrial Engineering and Management Sciences at Northwestern University. He is interested in the design and analysis of computer simulation experiments, particularly statistical efficiency, multivariate output analysis and input modeling. He is the simulation area editor for *Operations Research* and will be Program Chair for the 1997 Winter Simulation Conference.

Yu-Hui Tao is a senior software consultant working at a credit card company for various projects in risk management and marketing. He received his M.S. in 1989 and Ph.D. in 1995 from the Ohio state University. He is interested in system design and software development applying methodologies in Operations Research, Cognitive Engineering, and Computer Science.