SENATE:
A SOFTWARE SYSTEM
FOR EVALUATION OF SIMULATION RESULTS

by

Sandeep R. Somaiya

Project Report submitted to the faculty of the
Virginia Polytechinic Institute and State University
in partial fulfillment of the requirement for the degree of

Master of Science
in
Computer Science

APPROVED:

Osman Balci

Osman Balci, Chairman

John Roach

John Roach

Marc Abrams

Marc Abrams

April, 1993
Blacksburg, Virginia

SENATE: A SOFTWARE SYSTEM FOR EVALUATION OF SIMULATION RESULTS

by

Sandeep R. Somaiya

Osman Balci, Chairman

Computer Science

(ABSTRACT)

The purpose of this project described herein is to develop an advanced, modular Graphical User Interface (GUI) based tool for providing cost-effective, integrated, and automated support for assessing the credibility of simulation results using indicators.

The SENATE (A Software SystEm for EvaluatioN of SimulATion REsults) system prototype in this project meets this need by:

- providing a system that is easy to use, learn and maintain;
- providing general purpose indicators for each stage of the simulation life cycle;
- providing a system that will help the simulation project management to automate the means of capturing and retaining the expert assessment of the simulation study using indicators.

This report is meant to give the reader some background on the credibility assessment of simulation results as well as provide an overview of the various aspects of the project. The first chapter states the motivation, objectives, and scenario of the application. The second chapter provides an overview of the credibility and acceptability assessment of simulation results based on a literature survey. It also lists all the general purpose indicators that have been integrated into the system. Chapter 3 explains the design and implementation issues of the system. The fourth chapter explains the user interface and presents the user's guide. The final chapter suggests recommended areas of future research.

The system is implemented on a NeXT workstation using the NeXTSTEP Release 3 developer environment. SENATE has been successfully tested by three different users.

ACKNOWLEDGEMENTS

Many thanks are owed to Dr. Osman Balci for his undying support, guidance, and patience. This project would not have been possible without him.

I wish to thank Dr. Vinod Chachra and VTLS Inc. for the financial support; my coworkers Joe Derrick, Troy Gustafson, and Tom Woodard for all their help; my friends Alex and Mo for their constant encouragement; and my parents for all their support.

And finally, I wish to thank my hero and inspiration Steve Jobs for his vision.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1
INTRODUCTION

1.1 Motivation and Objectives

Since the advent of computers, simulation has emerged as one of the most powerful techniques for problem solving in many disciplines. As the problems grow larger, become more complex, and require more precise solutions than ever before, the use of simulation becomes more frequent.

In a simulation study, we work with a model of the problem rather than directly working with the problem itself. If the model does not possess a sufficiently accurate representation, we can easily have *junk input* and *junk output* [Balci 1987].

All simulation models are descriptive in nature; that is, a simulation model represents the behavior of a system without any value judgement on the *goodness* or *badness* of such behavior [Elmaghraby 1968]. Therefore, it is the responsibility of the simulationist to analyze and interpret the results of a simulation model.

Multifaceted and multidisciplinary knowledge and experience is required for a successful simulation study. A successful simulation study is one whose results are credible and are accepted and used by the decision makers (or sponsors). The results are assessed to be credible with the use of indicators (also called measures, scales, or factors) where appropriate. An indicator is an indirect measure of a concept and it can be measured directly [Nunnally 1978].

Subjectivity is and will always be a part of the credibility assessment for a simulation study. The reason for subjectivity is two-fold: modeling is an art and credibility assessment is situation dependent. A subjective, yet quite effective method for evaluating the acceptability of simulation results is peer-assessment, the assessment of the acceptability by a panel of

expert peers. Working together and sharing their knowledge, panel members measure the indicators and decide whether the simulation results will be accepted or used by the decision makers (or sponsor) [Balci 1987].

The objective of this project is to:

- implement an advanced, modular Graphical User Interface (GUI) based tool for a system that will help the simulation project management to automate the means of capturing and retaining the expert assessment of the simulation study using indicators;
- the GUI should be easy to use, easy to learn and easy to maintain;
- offer cost-effective, integrated, and automated support for assessing the credibility of simulation results;
- increase the efficiency and productivity of the expert peers;
- substantially decrease the credibility assessment time; and
- improve the confidence in the assessment by effectively assisting the expert peers during the assessment process.

The system that we have implemented is named SENATE (A Software SystEm for EvaluatioN of SimulATion REsults). Future work will involve building a knowledge-based system for processing the information that the SENATE system captures and retains. This knowledge-based system will be integrated into SENATE.

1.2 Scenario of System Application

To understand the application and significance of the system we need to understand the scenario under which the system will be applied. We assume that the approach to simulation model development used is the one suggested by Balci and Nance [Balci 1989] and propose the following scenario as an effective application scenario [Figure 1.1].

The organization that provides the funds for conducting the simulation study (e.g., Department of Defense) will be called the sponsors (or the decision makers). The

organization responsible for conducting the simulation study will be called the contractors. We assume that the sponsor hires a third party company to certify the credibility of the simulation study that was developed by the contractors. In order to assess the credibility of the results, instead of using conventional methods of employing experts and using techniques such as questionaires, forms, interviews etc., the third party will use the SENATE system. This system would help them avoid the conventional methods that are time consuming, expensive and inefficient.

The system would have built-in general purpose indicators that can be applied to any simulation study and can also be tailored to suit the particular application domain. The third party would independently appoint domain experts to create new application domain specific indicators. Then they would appoint a panel of independent peers or experts for assessing the credibility of the simulation study results by evaluating the indicators for the model.

1.3 Report Overview

This report is meant to give the reader some background on the credibility assessment of simulation results as well as provide an overview of the various aspects of the project undertaken by the author. The current chapter states the motivation, objectives, and scenario of application of the report. The second chapter provides an overview of the credibility and acceptability assessment of simulation results based on a literature survey by the author. It also lists all the general purpose indicators that have been integrated into the system. Chapter 3 explains the design and implementation issues of the system. The fourth chapter explains the user interface and presents the user's guide. The final chapter presents a summary of the work accomplished and suggests areas of future research.

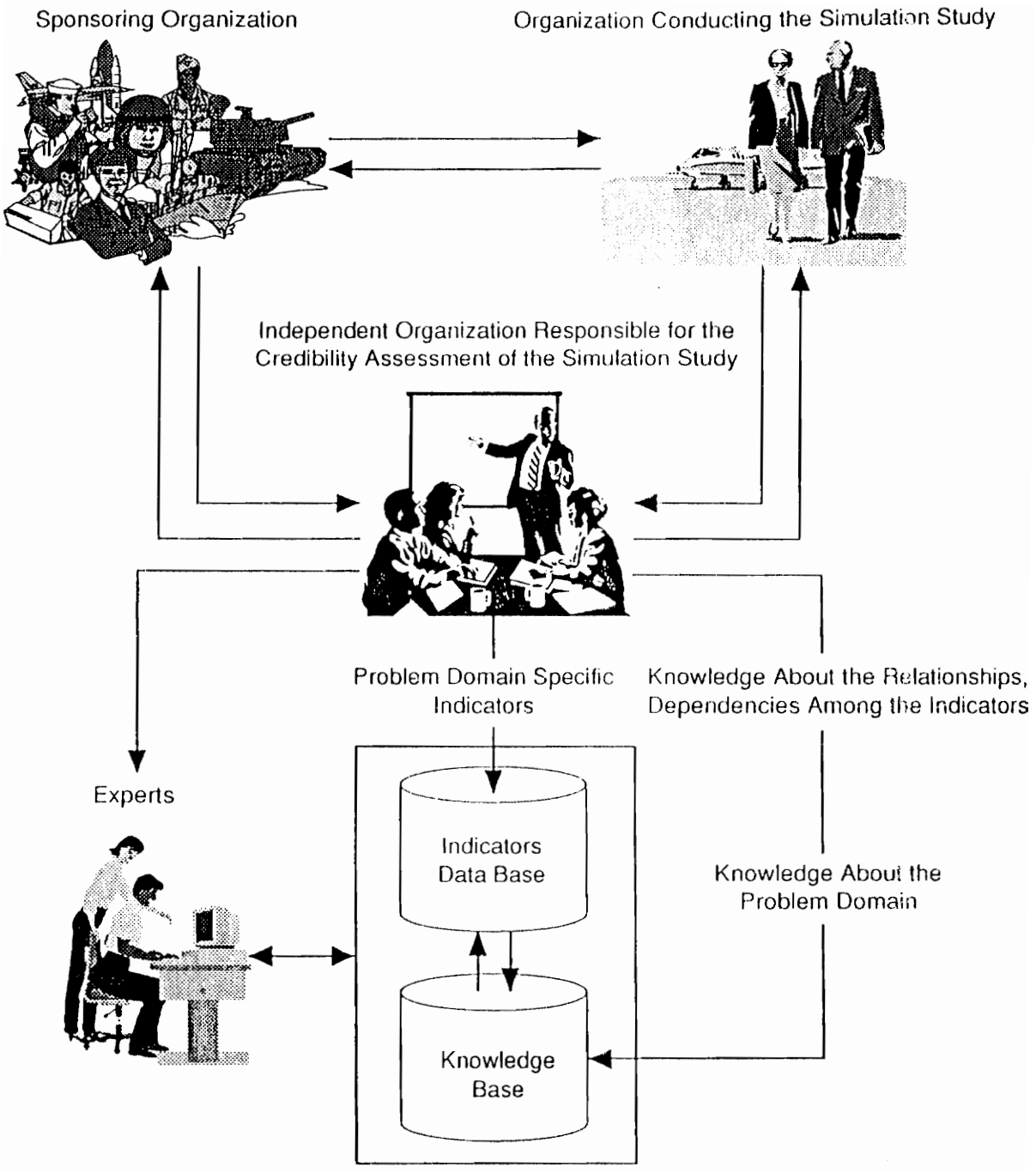


Figure 1.1 Scenario of System Application.

CHAPTER 2

ASSESSMENT OF THE CREDIBILITY AND ACCEPTABILITY OF SIMULATION RESULTS: AN OVERVIEW

In this chapter we present a comprehensive life cycle of a simulation study and the guidelines in conducting 10 processes, 10 phases, and 13 credibility assessment stages of the life cycle based on [Balci 1987]. In addition we also present the general purpose indicators that have been developed for assessing the credibility and acceptability of simulation results for each credibility assessment stage of the life cycle. Most of the material in this chapter is based on the work done by Balci [Balci 1987].

2.1 The Life Cycle of a Simulation Study

The following description of the life cycle, taken from [Balci 1987], is repeated here for the purpose of completeness.

The life cycle is composed of ten phases as shown in Figure 2.1. The ten phases are shown by oval symbols. The dashed arrows describe the processes which relate the phases to each other. The solid arrows refer to the Credibility Assessment Stages (CAS).

The life cycle should not be interpreted as strictly sequential. The sequential representation of the dashed arrows is intended to show the direction of development throughout the life cycle. The life cycle is iterative in nature and reverse transitions are expected.

The processes of the life cycle are:

Problem Formulation is the process by which the initially communicated problem is translated into a formulated problem sufficiently well defined to enable specific research action.

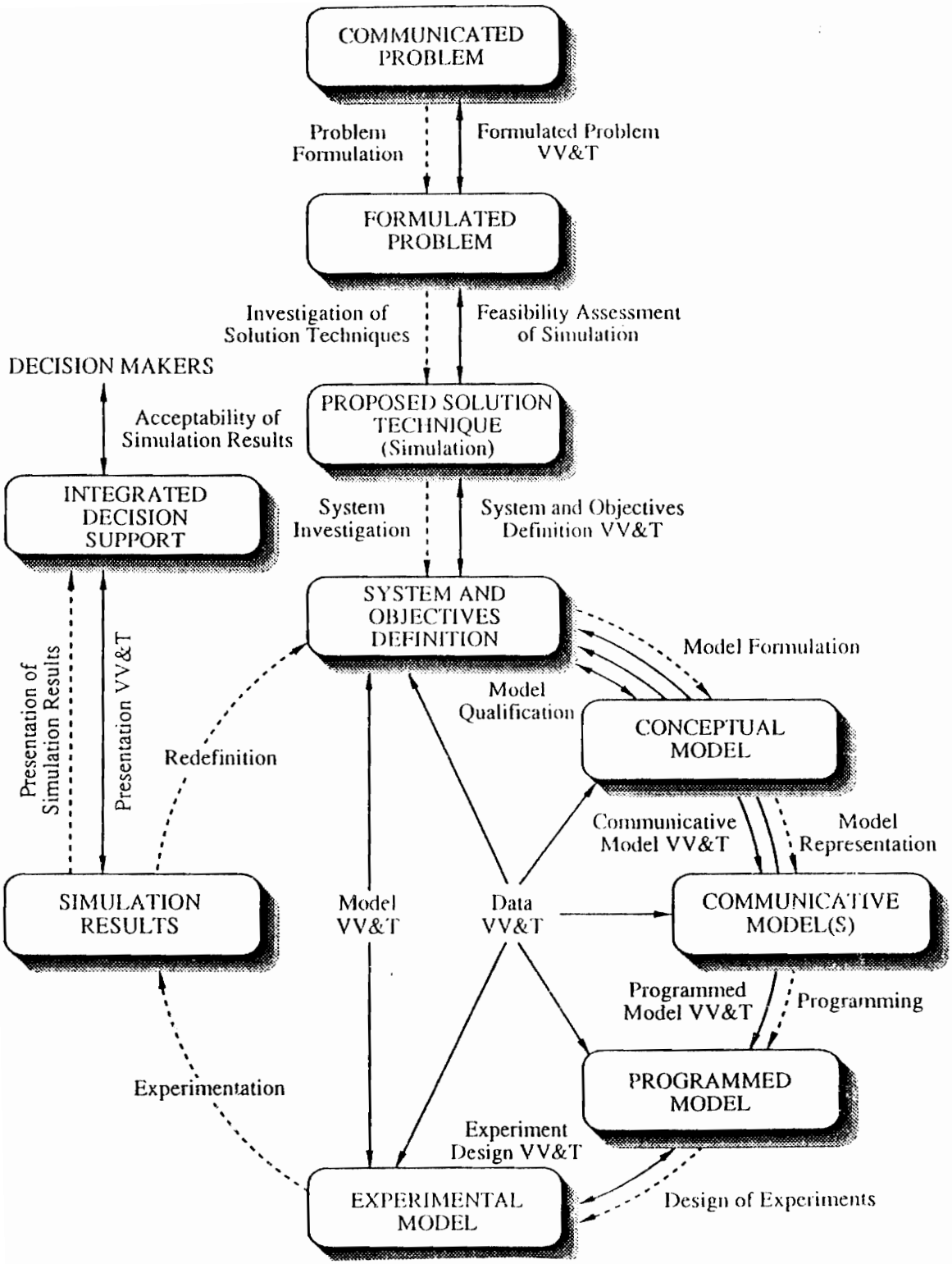


Figure 2.1 The Life Cycle of a Simulation Study.

Investigation of Solution Techniques involves all alternative techniques that can be used in solving the formulated problem should be identified.

The process of **System Investigation** involves the investigation of the characteristics of the system that contains the formulated problem. There are six major system characteristics: change, counter-intuitive behavior, drift to low performance, interdependency, and organization. In this process each characteristic should be examined with respect to the study objectives that are identified with the formulation of the problem.

Model Formulation is the process by which the conceptual model is envisioned to represent the system under study. The conceptual model which is formulated in the mind of the modeler.

Model Representation is the process of translating the conceptual model into a communicated model. A communicated model is "a model representation which can be represented to other humans, can be judged or compared against the system and the study objectives by more than one human" [Nance 1981].

The translation of a communicated model into a programmed model constitutes the process of **Programming**. A programmed model is an executable simulation model representation in a programming language.

The process of formulating a plan to gather the desired information at a minimal cost and to enable the analyst to draw valid inferences constitutes the process of **Design of Experiments**.

The process of Experimentation involves experimenting with the simulation model for a specific purpose. Some purposes of experimentation are comparison of different operating policies, evaluation of system behavior, sensitivity analysis, forecasting, optimization, and determination of functional relations. The process of **Experimentation** produces simulation results.

In the process of **Redefinition** we update the experimental model so that it represents the current form of the system, alter it for the purpose of obtaining another set of simulation

results, changing it for the purpose of maintenance, modify it for other use, or redefining a new system model for studying an alternative solution to the problem.

In the process of **Presentation of Simulation Results**, we interpret and present the simulation results to the decision makers for their acceptance and implementation.

2.2 Credibility Assessment Stages of the Life Cycle

A successful simulation study is the one whose results are credible and are accepted and used by the decision makers (or sponsor of the study). It is crucial that we assess the credibility of each process as we progress in the life cycle. Since a model is an abstraction of the reality, we cannot talk about its absolute accuracy. Credibility, quality, validity, and verity are measures that are assessed with respect to the study objectives for which the model is intended.

A subjective, yet quite effective method of evaluating the acceptability of model (simulation) results is peer assessment, the assessment of the acceptability by a panel of expert peers. This panel should be composed of

- people who have expert knowledge of the system under study,
- expert modelers,
- expert simulation analysts, and
- people with extensive experience with simulation projects.

The panel examines the overall study based upon the project team's presentation and detailed study of documentation. Working together and sharing their knowledge among each other, panel members measure the indicators shown by the leaves of the tree in Figure 2.2 which assists in explaining the hierarchy of CASs. A branch of the tree represents a CAS except the branch of "other indicators" of experimental model quality.

An indicator is an indirect measure of a concept. It can be decomposed into other indicators. The ones at the base level should be directly measurable. The indicators (leaves)

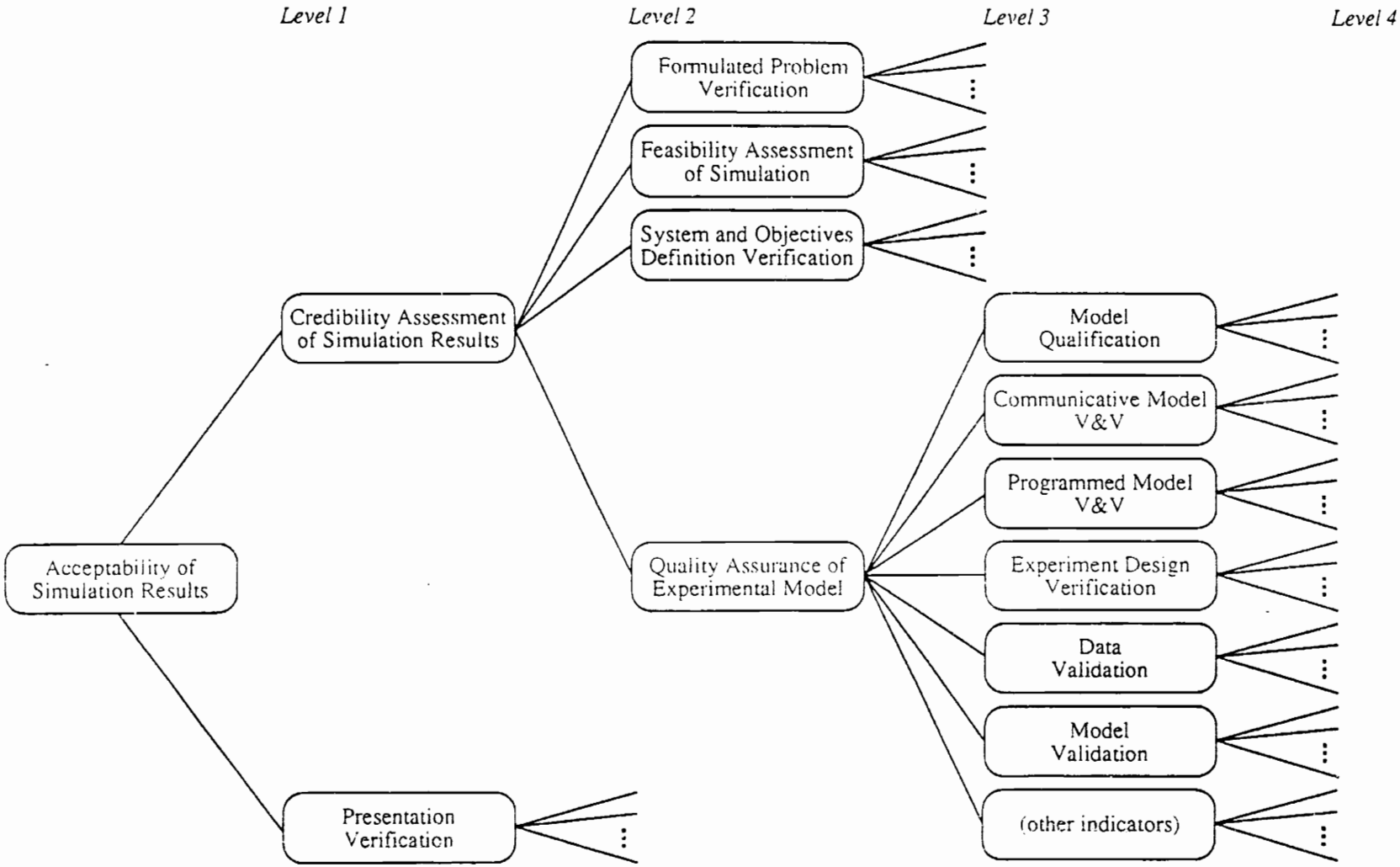


Figure 2.2 A Hierarchy of Credibility Assessment Stages for Evaluating the Acceptibility of Model (Simulation) Results.

are presented in the following subsections. The kth indicator out of N_{ij} ones corresponding to the jth branch at level i [Figure 2.2] is measured with a score, S_{ijk}, out of 100 and is weighted with W_{ijk}, a fractional value between 0 and 1, according to its importance. The following constraint must be satisfied:

$$\sum_{k=1}^{N_{ij}} W_{ijk} = 1, \qquad (i,j) = \{(1,2),(2,1),(2,2),(2,3),(3,1),(3,2),...,(3,7)\}.$$

Thus, a score for the jth branch at level i, S_{ij}, is calculated on a scale from 0 to 100 where 0 represents "not credible at all" and 100 means "sufficiently credible."

$$S_{ij} = \sum_{k=1}^{N_{ij}} W_{ijk} S_{ijk}, \qquad (i,j) = \{(1,2),(2,1),(2,2),(2,3),(3,1),(3,2),...,(3,7)\}.$$

In addition to weighting the indicators, panel members can also weight the branches based on experience and training. For example, model validation branch should be given higher weight than the other branches at level 3, if it is possible to validate the model objectively using the real system data under all experimental conditions of interest. On the other hand, if the model represents a nonexistent system or a future-oriented situation in which the past is not a good predictor of the future, higher weight should be given to other branches at level 3.

Assume that the Wij denotes the weight for the jth branch at level i. W_{ij} is a fractional value between 0 and 1 where 0 represents "not critical at all" and 1 indicates "extremely critical." W_{ij}'s are specified with the following constraints:

$$\sum_{j=1}^{2i} W_{ij} = 1, \qquad i = 1,2 \quad \text{and} \quad \sum_{j=1}^{7} W_{3j} = 1.$$

Thus, a credibility score for the quality assurance branch is calculated as

$$S_{24} = \sum_{j=1}^{7} W_{3j} S_{3j}.$$

Similarly, S_{11} is computed and an overall score, S (= $W_{11}S_{11}$ + $W_{12}S_{12}$), is obtained as a

value on a scale from 0 to 100.

The higher the overall score the more confidence we gain for the acceptability of model results. However, even a perfect score should not guarantee that the results will be accepted and used by the decision makers; because, acceptability is an attribute of the decision maker not an attribute of the simulation study. Perfect results may be rejected due to lack of credibility of the institution performing the study or due to a political reason. Nevertheless, the objective should be to increase the confidence as much as possible. A higher overall score may result in rejection or an error of type II. Type II error is committed when the study results are accepted when in fact they are not sufficiently credible.

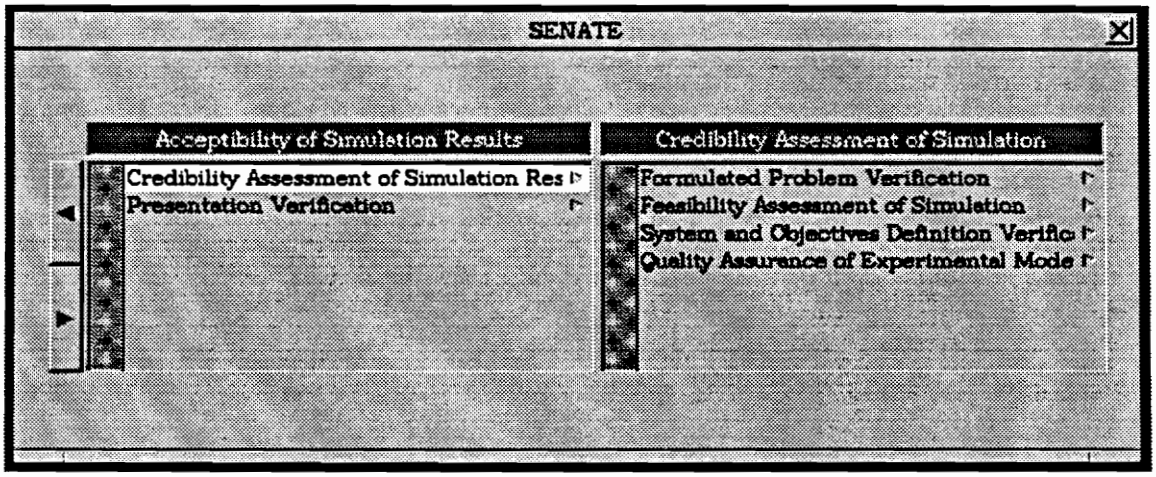


Figure 2.3 Main Browser showing the Credility Assessment Stages.

In the following subsections we present the Credibility Assessment Stages (CASs) of the simulation life cycle and the general purpose indicators that we have included in the system for each CAS. SENATE permits domain or problem specific indicators to be added to these general purpose indicators for the simulation project at hand.

2.2.1 Formulated Problem Verification [Balci & Nance 1987]

Substantiation that the formulated problem contains the actual problem in its entirety and is sufficiently well structured to permit derivation of a sufficiently credible solution is called formulated problem verification [Balci and Nance 1985] and is presented below:

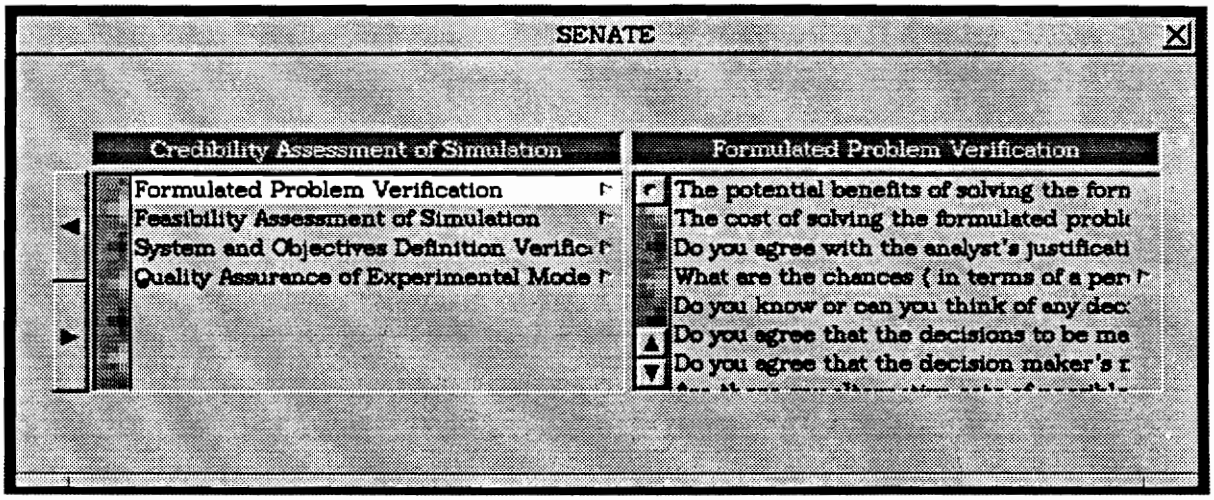


Figure 2.4 Formulated Problem Verification CAS

The general purpose indicators that we have included in SENATE for this stage of the life cycle are taken from [Balci and Nance 1985] and presented below:

1. The potential benefits of solving the formulated problem are:

0. Overestimated a lot
25. Overestimated
50. Estimated close enough
75. Underestimated
100. Underestimated a lot

2. The cost of solving the formulated problem is:

0. Overestimated a lot
25. Overestimated
75. Underestimated
100. Underestimated a lot

50. Estimated close enough

3. Do you agree with the analyst's justification that the formulated problem is worthwhile to solve?

0. Strongly agree
33. Agree
66. Disagree
100. Strongly disagree

4. What are the chances (in terms of a percentage) that the actual problem is not completely identified due to the possibility that:

[Evaluate the indicators under this indicator in the hierarchy]

4.1 What are the chances that the actual problem is not completely identified due to the possibility that people might have personalized problems?

[Score on a scale of 1-100]

4.2 What are the chances that the actual problem is not completely identified due to the possibility that information showing that a problem exists might not have been r evealed?

[Score on a scale of 1-100]

4.3 What are the chances that the actual problem is not completely identified due to the possibility that the problem context is too complex for the analyst to comprehend?

[Score on a scale of 0-100]

4.4 What are the chances that the actual problem is not completely identified due to the possibility that root problems might have arisen in context with which people have had no experience?

[On a scale of 0-100]

4.5 What are the chances that the actual problem is not completely identified due to the possibility that cause and effect may not be closely related within the problem context?

[On a scale of 0-100]

4.6 What are the chances that the actual problem is not completely identified due to the possibility that the analyst might have been unable to distinguish between facts and opinions ?

[On a scale of 0-100]

4.7 What are the chances that the actual problem is not completely identified due to the possibility that the analyst might have been misguided deliberately or accidently?

[On a scale of 0-100]

4.8 What are the chances that the actual problem is not identified due to the possibility that the level of extraction of problem context was insufficiently detailed:

[On a scale of 0-100]

4.9 What are the chances that the actual problem is not completely identified due to the possibility that the problem boundary was insufficient to include the entire problem:

[On a scale of 0-100]

4.10 What are the chances that the actual problem is not completely identified due to the possibility that inadequate standards or definition of desired conditions exist:

[On a scale of 0-100]

4.11 What are the chances that the actual problem is not completely identified due to the possibility that the root causes might be time dependent:

[On a scale of 0-100]

4.12 What are the chances that the actual problem is not completely identified due to the possibility that a root cause might have been masked by the emphasis on another:

[On a scale of 0-100]

4.13 What are the chances that the actual problem is not completely defined due to the possibility that invalid information might have been used:

[On a scale of 0-100]

4.14 What are the chances that the actual problem is not completely defined due to the possibility that invalid data might have been used:

[On a scale of 0-100]

4.15 What are the chances that the actual problem is not completely defined due to the possibility that assumptions might have concealed root causes:

[On a scale of 0-100]

4.16 What are the chances that the actual problem is not completely defined due to the possibility that resistance might have occurred from people suspicious of change:

[On a scale of 0-100]

4.17 What are the chances that the actual problem is not completely defined due to the possibility that the problem was formulated under the influence of a solution

technique:

[On a scale of 0-100]

4.18 What are the chances that the actual problem is not completely defined due to the possibility that the real objectives might have been hidden accidently, unconsciously, or deliberately:

[On a scale of 0-100]

4.19 What are the chances that the actual problem is not completely defined due to the possibility that root causes might be present in other unidentified systems, frameworks, or structures:

[On a scale of 0-100]

4.20 What are the chances that the actual problem is not completely defined due to the possibility that the formulated problem may be out of date:

[On a scale of 0-100]

5. Do you know or can you think of any decision makers, other than the ones identified by the analyst, who might be aided by the solution of the problem?

50. NO 100. YES

If, YES list them using the NOTES option.

6. Do you agree that the decisions to be made by the decision makers are completely and correctly identified?

0. Strongly agree 75. Disagree

25. Agree 100. Strongly disagree

7. Do you agree that the decision maker's needs for making the decisions are completely and correctly identified?

0. Strongly agree
25. Agree
75. Disagree
100. Strongly disagree

8. Are there any alternative sets of possible outcomes generated by the analyst that you believe, are unacceptable to the decision makers or cannot be implemented?

50. YES
100. NO

9. Do you know or can you think of any other alternative sets of possible outcomes which would be acceptable to the decision makers?

50. YES
100. NO

If YES, use NOTES option; and list them and explain each in detail

10. Do you know or think of any relevant decision makers, other than the ones identified by the analyst, who may influence the acceptability of any one of the alternative sets of possible outcomes?

50. YES
100. NO

If YES, list them in NOTES.

11. Do you know or can you think of any relevant decision makers, other than the ones identified by the analyst, who may cause rejection of any one of the alternative sets of possible outcomes by way of strong objections or counteractions against its implementation?

50. YES
100. NO

If YES, list them.

12. What are the chances (in terms of a percentage) that the *i*th (i=1,2,3,...,l) alternative set of possible outcomes is rejected due to the possibility that.....

[Evaluate the 8 child indicators under this parent indicator on a scale of 0-100]

12.1 What are the chances in terms of a percentage that the *i*th alternative set of possible outcomes is rejected due to the possibility that a key decision maker to whom the *i*th alternative is not acceptable may not have been identified:

12.2 What are the chances in terms of a percentage that the *i*th alternative set of possible outcomes is rejected due to the possibility that the *i*th alternative might have been unacceptable due to the substantial changes occurred in the problem context:

12.3 What are the chances that the *i*th alternative set of possible outcomes is rejected due to the possibility that the analyst might have failed to interact with the decision makers during the process of problem formulation:

12.4 What are the chances in terms of a percentage that the *i*th alternative set of possible outcomes is rejected due to the possibility that an important element of the problem context might have been excluded from the *i*th alternative:

12.5 What are the chances in terms of a percentage that the *i*th alternative set of possible outcomes is rejected due to the possibility that an important alternative set of possible outcomes might have been ignored:

12.6 What are the chances in terms of a percentage that the *i*th alternative set of possible outcomes is rejected due to the possibility of strong objections or counteractions against its implementation:

12.7 What are the chances in terms of a percentage that the *i*th alternative set of possible outcomes is rejected due to the possibility of its high cost of implementation:

12.8 What are the chances in terms of a percentage that the *i*th alternative set of possible outcomes is rejected due to the possibility of its unacceptability to a key decision maker:

13. Do you know or can you think of any other constraints which should have been identified by the analyst?

50. YES 100. NO

If YES, choose the NOTES option and list them.

14. Are there any incorrect or irrelevant constraints?

50. YES 100. NO

If YES, list them using the NOTES option.

15. Are there any constraints which make the formulated problem infeasible to solve?

0. YES 100. NO

If YES, list them using the NOTES option.

16. How well do the objective function values represent the attainment of objectives?

0. Excellent 66. Fair

33. Good 100. Poor

17. Do you know or can you think of any relevant decision makers, other than the ones identified by the analyst, who would not accept the objective function(s)?

0. YES 100. NO

If YES, list them using the NOTES option.

18. Do all the decision makers involved accept the objective function(s)?

50. YES 100. NO

If YES, list them using the NOTES option

19. How clearly are the objective functions stated?

0. Very clearly 66. Unclearly

33. Clearly 100.Very unclearly

20. Do you believe any objectives are inconsistent, ambiguous, or conflicting in any way?

50. YES 100. NO

If YES, list them and explain them in detail.

21. How realistic are the objectives?

0. Very realistic 66. Unrealistic

33. Realistic 100. Very realistic

22. Are there any priorities specified for the case where only some of the objectives are achievable?

0. YES 100. NO

23. Do you know or can you think of any relevant decision makers whose objectives are conflicting with any one of those specified?

50. YES 100. NO

If YES, list them using the NOTES option.

24. In case of multiple objectives, do you agree with the way the objectives are weighted?

50. YES 100. NO

If YES, list them using the NOTES option.

25. Do you agree that the stated objectives are the real objectives of the decision makers involved?

0. Strongly agree 66. Disagree

33. Agree 100. Strongly disagree

26. Do you know or can you think of any associated objective which is disguised or hidden either accidently, unconsciously, or deliberately?

50. YES 100. NO

If YES, list them using the NOTES option.

27. How often do the stated objectives change?

0. Always 75. Seldom

25. Usually 100. Never

50. Sometimes

28. How sufficient are the stated performance measures for attaining the objectives or for making the decisions?

0. Very sufficient 66. Insufficient

33. Sufficient 100. Very sufficient

29. Do all the decision makers involved accept the performance measure(s)?

50. YES 100. NO

If NO, list the ones unacceptable with the respective decision makers.

30. Do you know or can you think of any relevant decision makers, other than the ones identified by the analyst, who would not accept the performance measure(s)?

50. YES 100. NO

If YES, list them using the NOTES option.

31. Are there any sources of data and information used by the analyst that you believe to be unbelievable?

50. YES 100. NO

If YES, list them.

32. Are there any data and information used by the analyst that you believe to be out of date or need to be updated?

50. YES 100. NO

If YES, list them.

33. Are there any data and information which you believe to be not sufficiently accurate?

50. YES 100. NO

If YES, list them using the NOTES option.

34. Are there any invalid assumptions?

50. YES 100. NO

If YES, list them and give a rationale for each of them [using the Notes option].

35. Are there any invalid inferences or conclusions drawn by the analyst?

50. YES 100. NO

If YES, list them and give a rationale for each of them.

36. How clearly are the requirements for the certification of credibility of the results stated?

0. Very clearly 66. Unclearly

33. Clearly 100. Very unclearly

37. Do you know or can you think of any relevant people, other than the ones identified by the analyst, who may influence the certification of the credibility of the results?

50. YES 100. NO

If YES, list them.

38. Do you know or can you think of any certification requirements appropriate to specify in the formulated problem?

50. YES 100. NO

If YES, list them using the NOTES option.

2.2.2 Feasibility Assessment of Simulation

Feasibility Assessment involves finding whether the benefits and cost of simulation have been estimated correctly or do the potential benefits of the simulation solution justify the cost of obtaining it or can the necessary resources be secured or is it possible to solve the problem using simulation within the specified time period. These questions are the indicators of the feasibility of simulation.

The general purpose indicators that we have included in SENATE for this stage of the life cycle are taken from [Balci 1987] and presented below:

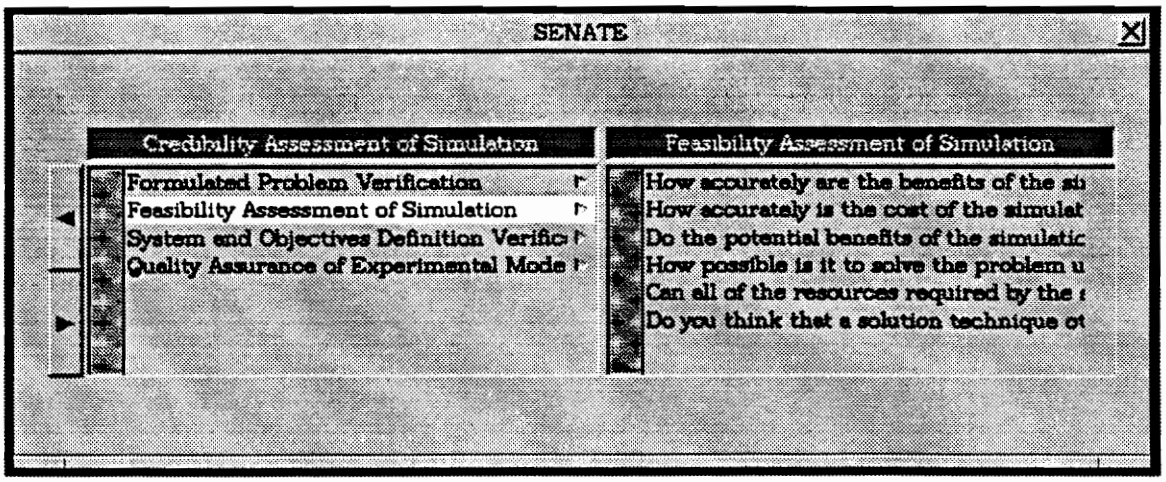


Figure 2.5 Feasibility Assessment of Simulation CAS

1. How accurately are the benefits of the simulation solution estimated?

0. Absolutely inaccurately 100. Absolutely accurately

2. How accurately is the cost of the simulation solution estimated?

0. Absolutely inaccurately 100. Absolutely accurately

3. Do the potential benefits of the simulation solution justify the cost of obtaining it?

0. Absolutely no 100. Absolutely yes

4. How possible is it to solve the problem using simulation within the time limit specified?

0. Absolutely impossible 100. Absolutely possible

5. Can all of the resources required by the simulation project (e.g., personnel, equipment, access to classified information) be secured?

0. Absolutely no 100. Absolutely yes

6. Do you think that a solution technique other than simulation must be used to solve the problem?

0. Absolutely no 100. Absolutely yes

2.2.3 System and Objectives Definition Verification

We should justify that the system characteristics are identified and the study objectives are explicitly defined with sufficient accuracy.

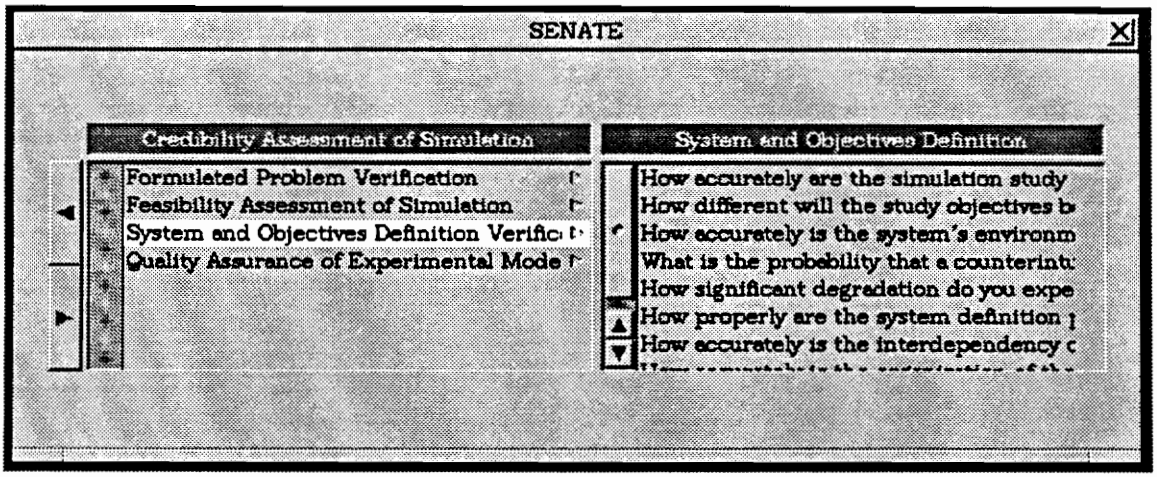


Figure 2.6 Systems and Objectives Definition Verification CAS

The general purpose indicators that we have included in SENATE for this stage of the life cycle are taken from [Balci 1987] and presented below::

1. How accurately are the simulation study objectives identified?

0. Absolutely inaccurately 100. Absolutely accurately

2. How different will the study objectives be at the conclusion of the simulation study? (Over the course of the simulation study, management objectives and policies may

change resulting in changes in the study objectives.)

0. Totally different 100. Exactly the same

3. How accurately is the system's environment (boundary) identified?

0. Absolutely inaccurately 100. Absolutely accurately

4. What is the probability that a counterintuitive system behavior can significantly invalidate the system and objectives definition?

0. 0% Probability 100. 100% Probability

5. How significant degradation do you expect in the system performance during the course of the simulation study?

0. No degradation whatsoever 100. Extremely high degradation

6. How properly are the system definition periodic updates (required due to expected degradation of the system performance) scheduled during the course of the simulation study?

0. Absolutely improperly 100. Absolutely properly

7. How accurately is the interdependency of the system characterized?

0. Absolutely inaccurately 100. Absolutely accurately

8. How accurately is the organization of the system characterized?

0. Absolutely inaccurately 100. Absolutely accurately

2.2.4 Model Qualification

Model Qualification deals with the justification that all assumptions made are appropriate and the conceptual model provides an adequate representation of the system with respect to the study objectives.

The general purpose indicators that we have included in SENATE for this stage of the life cycle are taken from [Balci 1987] and presented below:. Figure 2.7 shows the main browser with the Model Qualification CAS general purpose indicators.

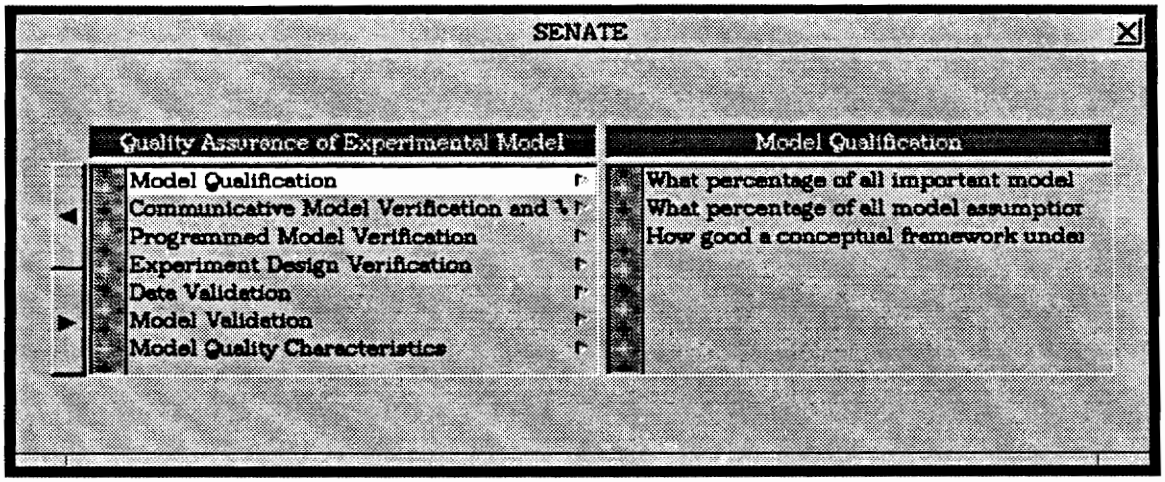


Figure 2.7 Model Qualification CAS

1. What percentage of all important model assumptions are explicitly identified and documented?

   0. 0% 100. 100%

2. What percentage of all model assumptions are acceptably substantiated?

   0. 0% 100. 100%

3. How good a conceptual framework under the guidance of which the model is specified?

0. Absolutely bad 100. Absolutely good

2.2.5 Communicative Model Verification and Validation

In this stage, we confirm the adequacy of the communicative model to provide an acceptable level of agreement for the domain of intended application. Domain of Intended Application is the prescribed conditions for which the model is intended to match the system under study. Level of Agreement is the required correspondence between the model and the system, consistent with the domain of intended application and the study objectives.

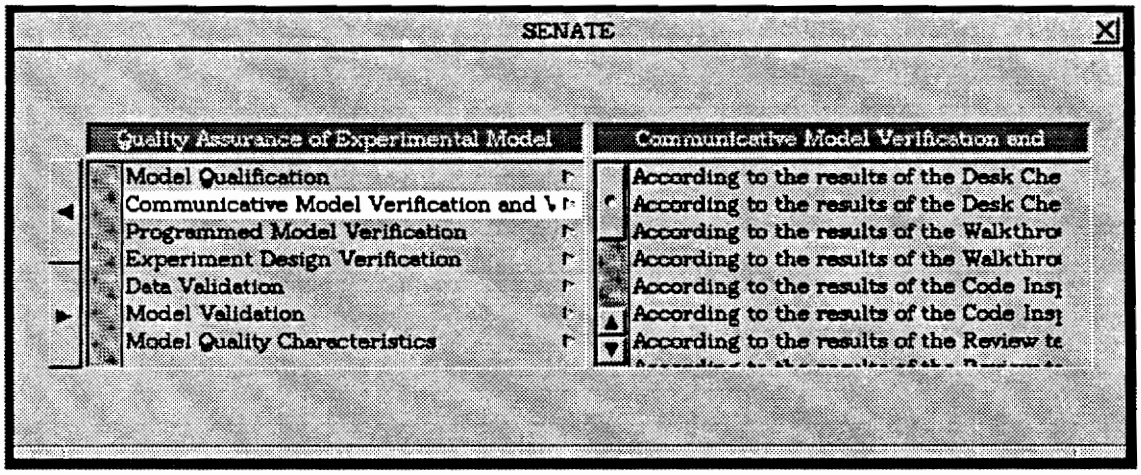


Figure 2.8 Communicative Model Verification and Validation CAS

Communicative Model Verification and Validation can be conducted by using one or more informal and static analysis techniques (e.g., desk checking, walkthrough, code inspection, review, audit, structural analysis, data flow analysis, etc.) described in [Balci 1987].

The general purpose indicators that we have included in SENATE for this stage of the life

cycle are taken from [Balci 1987] and presented below:

1. According to the results of the Desk Checking testing technique (if not used, score zero), how accurately is the conceptual model translated into the communicative model?

   0. Absolutely inaccurately 100. Absolutely accurately

2. According to the results of the Desk Checking testing technique (if not used, score zero), how accurately does the communicative model represent the system under study?

   0. Absolutely inaccurately 100. Absolutely accurately

3. According to the results of the Walkthrough testing technique (if not used, score zero), how accurately is the conceptual model translated into the communicative model?

   0. Absolutely inaccurately 100. Absolutely accurately

4. According to the results of the Walkthrough testing technique (if not used, score zero), how accurately does the communicative model represent the system under study?

   0. Absolutely inaccurately 100. Absolutely accurately

5. According to the results of the Code Inspection testing technique (if not used, score zero), how accurately is the conceptual model translated into the communicative model?

   0. Absolutely inaccurately 100. Absolutely accurately

6. According to the results of the Code Inspection testing technique (if not used, score zero), how accurately does the communicative model represent the system under

study?

0. Absolutely inaccurately 100. Absolutely accurately

7. According to the results of the Review testing technique (if not used, score zero), how accurately is the conceptual model translated into the communicative model?

0. Absolutely inaccurately 100. Absolutely accurately

8. According to the results of the Review testing technique (if not used, score zero), how accurately does the communicative model represent the system under study?

0. Absolutely inaccurately 100. Absolutely accurately

9. According to the results of the Audit testing technique (if not used, score zero), how accurately is the conceptual model translated into the communicative model?

0. Absolutely inaccurately 100. Absolutely accurately

10. According to the results of the Audit testing technique (if not used, score zero), how accurately does the communicative model represent the system under study?

0. Absolutely inaccurately 100. Absolutely accurately

11. According to the results of the Structural Analysis testing technique (if not used, score zero), how accurately is the conceptual model translated into the communicative model?

0. Absolutely inaccurately 100. Absolutely accurately

12. According to the results of the Structural Analysis testing technique (if not used, score zero), how accurately does the communicative model represent the system under study?

0. Absolutely inaccurately 100. Absolutely accurately

13. According to the results of the Consistency Checking testing technique (if not used, score zero), how accurately is the conceptual model translated into the communicative model?

0. Absolutely inaccurately 100. Absolutely accurately

14. According to the results of the Consistency Checking testing technique (if not used, score zero), how accurately does the communicative model represent the system under study?

0. Absolutely inaccurately 100. Absolutely accurately

2.2.6 Programmed Model Verification and Validation

There are six techniques : informal, static, dynamic, symbolic, constraint, and formal analysis techniques. These techniques are applicable for Programmed Model Verification and Validation [Balci 1987].

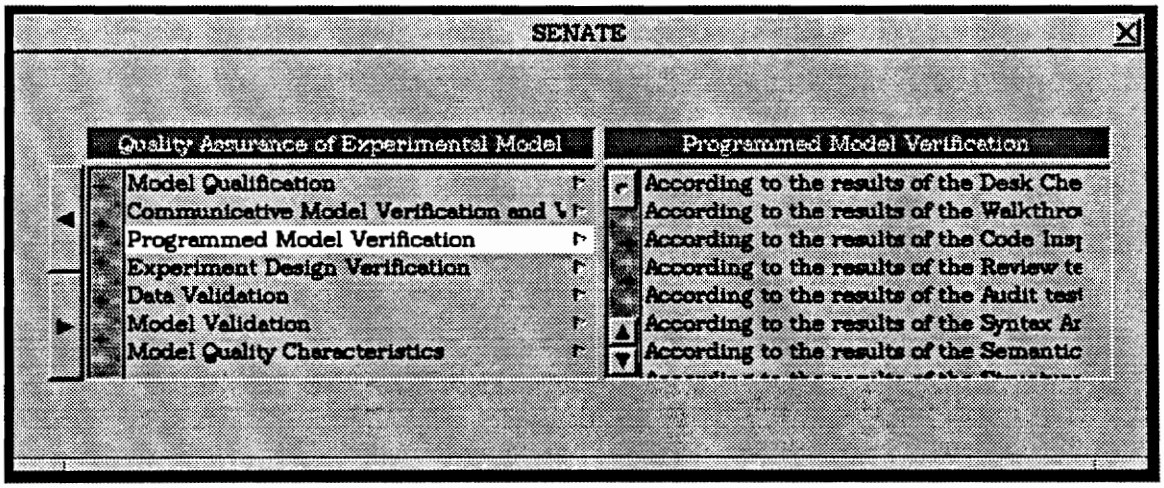


Figure 2.9 Programmed Model Verification and Validation CAS

The general purpose indicators that we have included in SENATE for this stage of the life cycle are taken from [Balci 1987] and presented below:

1. According to the results of the Desk Checking testing technique (if not used, score zero), how accurately is the communicative model translated into the programmed model?

   0. Absolutely inaccurately 100. Absolutely accurately

2. According to the results of the Walkthrough testing technique (if not used, score zero), how accurately is the communicative model translated into the programmed model?

   0. Absolutely inaccurately 100. Absolutely accurately

3. According to the results of the Code Inspection testing technique (if not used, score zero), how accurately is the communicative model translated into the programmed model?

   0. Absolutely inaccurately 100. Absolutely accurately

4. According to the results of the Review testing technique (if not used, score zero), how accurately is the communicative model translated into the programmed model?

   0. Absolutely inaccurately 100. Absolutely accurately

5. According to the results of the Audit testing technique (if not used, score zero), how accurately is the communicative model translated into the programmed model?

   0. Absolutely inaccurately 100. Absolutely accurately

6. According to the results of the Syntax Analysis testing technique (if not used, score zero), how accurately is the communicative model translated into the programmed

model?

0. Absolutely inaccurately 100. Absolutely accurately

7. According to the results of the Semantic Analysis testing technique (if not used, score zero), how accurately is the communicative model translated into the programmed model?

0. Absolutely inaccurately 100. Absolutely accurately

8. According to the results of the Structural Analysis testing technique (if not used, score zero), how accurately is the communicative model translated into the programmed model?

0. Absolutely inaccurately 100. Absolutely accurately

9. According to the results of the Data Flow Analysis testing technique (if not used, score zero), how accurately is the communicative model translated into the programmed model?

0. Absolutely inaccurately 100. Absolutely accurately

10. According to the results of the Consistency Checking testing technique (if not used, score zero), how accurately is the communicative model translated into the programmed model?

0. Absolutely inaccurately 100. Absolutely accurately

11. According to the results of the Top-Down Testing technique (if not used, score zero), how accurately is the communicative model translated into the programmed model?

0. Absolutely inaccurately 100. Absolutely accurately

12. According to the results of the Bottom-Up Testing technique (if not used, score zero), how accurately is the communicative model translated into the programmed model?

0. Absolutely inaccurately 100. Absolutely accurately

13. According to the results of the Black-Box Testing technique (if not used, score zero), how accurately is the communicative model translated into the programmed model?

0. Absolutely inaccurately 100. Absolutely accurately

14. According to the results of the White-Box Testing technique (if not used, score zero), how accurately is the communicative model translated into the programmed model?

0. Absolutely inaccurately 100. Absolutely accurately

15. According to the results of the Stress Testing technique (if not used, score zero), how accurately is the communicative model translated into the programmed model?

0. Absolutely inaccurately 100. Absolutely accurately

16. According to the results of the Execution Tracing testing technique (if not used, score zero), how accurately is the communicative model translated into the programmed model?

0. Absolutely inaccurately 100. Absolutely accurately

17. According to the results of the Execution Monitoring testing technique (if not used, score zero), how accurately is the communicative model translated into the programmed model?

0. Absolutely inaccurately 100. Absolutely accurately

18. According to the results of the Execution Profiling testing technique (if not used, score zero), how accurately is the communicative model translated into the programmed model?

0. Absolutely inaccurately 100. Absolutely accurately

19. According to the results of the Symbolic Debugging testing technique (if not used, score zero), how accurately is the communicative model translated into the programmed model?

0. Absolutely inaccurately 100. Absolutely accurately

20. According to the results of the Regression Testing technique (if not used, score zero), how accurately is the communicative model translated into the programmed model?

0. Absolutely inaccurately 100. Absolutely accurately

21. According to the results of the Symbolic Execution testing technique (if not used, score zero), how accurately is the communicative model translated into the programmed model?

0. Absolutely inaccurately 100. Absolutely accurately

22. According to the results of the Path Analysis testing technique (if not used, score zero), how accurately is the communicative model translated into the programmed model?

0. Absolutely inaccurately 100. Absolutely accurately

23. According to the results of the Cause-Effect Graphing testing technique (if not used, score zero), how accurately is the communicative model translated into the

programmed model?

0. Absolutely inaccurately 100. Absolutely accurately

24. According to the results of the Partition Analysis testing technique (if not used, score zero), how accurately is the communicative model translated into the programmed model?

0. Absolutely inaccurately 100. Absolutely accurately

25. According to the results of the Assertion Checking testing technique (if not used, score zero), how accurately is the communicative model translated into the programmed model?

0. Absolutely inaccurately 100. Absolutely accurately

26. According to the results of the Inductive Assertion testing technique (if not used, score zero), how accurately is the communicative model translated into the programmed model?

0. Absolutely inaccurately 100. Absolutely accurately

27. According to the results of the Boundary Analysis testing technique (if not used, score zero), how accurately is the communicative model translated into the programmed model?

0. Absolutely inaccurately 100. Absolutely accurately

2.2.7 Experiment Design Verification

In this stage the design of the experiments can be verified by measuring the indicators for this stage: Are the algorithms used for the random variate generation theoretically accurate?

Are the random variate generation algorithms translated into executable code accurately? How well is the random number generator tested? Are the appropriate statistical techniques implemented to design and analyze the simulation experiments? How well are the underlying assumptions satisfied? Is the problem of the initial transient appropriately addressed? For comparison studies, are identical experimental conditions replicated correctly for each of the alternative operating policies compared?

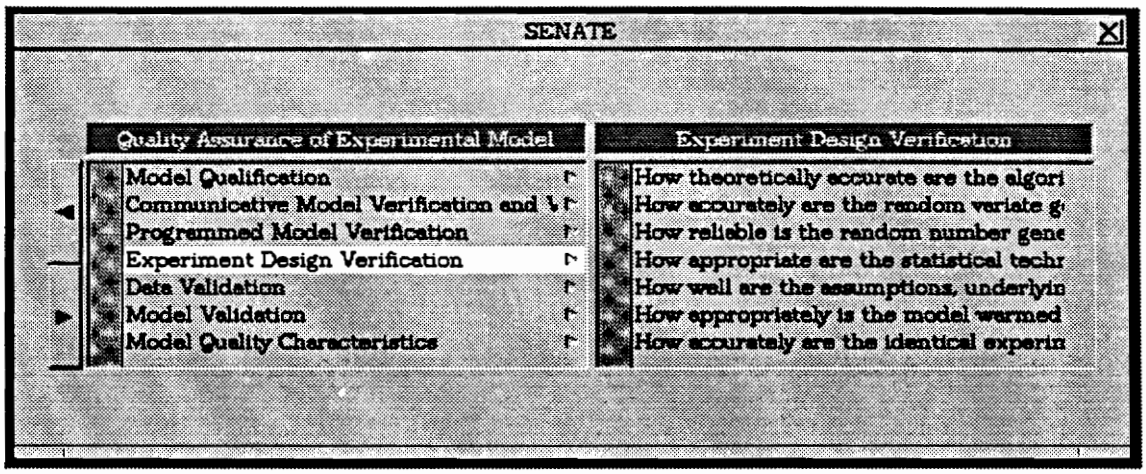


Figure 2.10 Experiment Design Verification CAS

The general purpose indicators that we have included in SENATE for this stage of the life cycle are taken from [Balci 1987] and presented below:

1. How theoretically accurate are the algorithms used for random variate generation?

0. Absolutely inaccurate 100. Absolutely accurate

2. How accurately are the random variate generation algorithms translated into xecutable code?

0. Absolutely inaccurately 100. Absolutely accurately

3. How reliable is the random number generator?

0. Absolutely unreliable 100. Absolutely reliable

4. How appropriate are the statistical techniques used to design and analyze the simulation experiments?

0. Absolutely inappropriate 100. Absolutely appropriate

5. How well are the assumptions, underlying the statistical techniques used, satisfied?

0. Not satisfied at all 100. Perfectly satisfied

6. How appropriately is the model warmed up to remove the effects of the initial transient or start up phase?

0. Absolutely inappropriately 100. Absolutely appropriately

7. How accurately are the identical experimental conditions replicated for each of the alternative operating policies compared? (If not a comparison study, remove this indicator)

0. Absolutely inaccurately 100. Absolutely accurately

2.2.8 Data Validation

In this stage, we confirm that the data used throughout the model development phases are accurate, complete, unbiased, and appropriate in their original and transformed forms.

The general purpose indicators that we have included in SENATE for this stage of the life cycle are taken from [Balci 1987] and presented below:

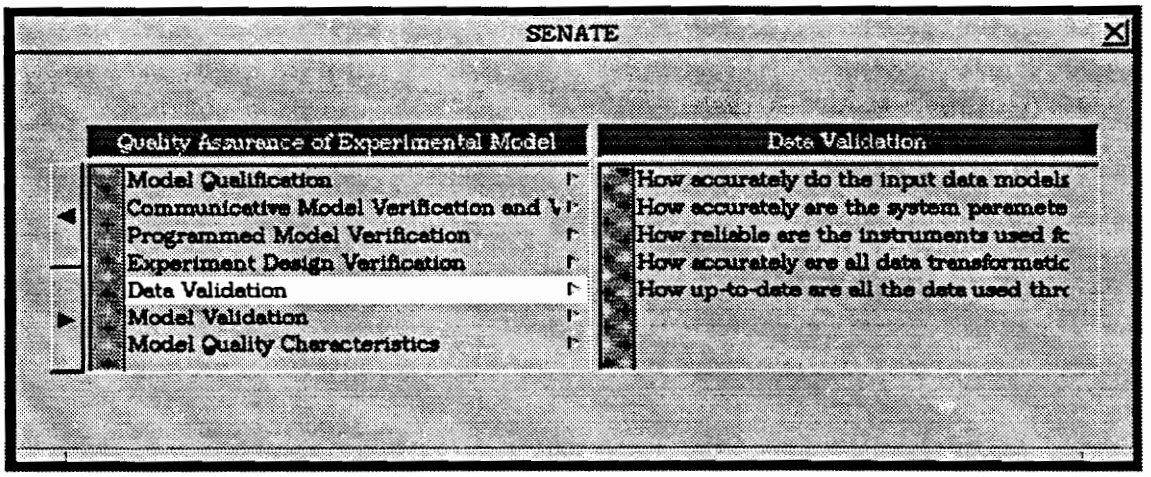


Figure 2.11 Data Validation CAS

1. How accurately do the input data models represent the input of the system under study?

   0. Absolutely inaccurately 100. Absolutely accurately

2. How accurately are the system parameter values identified, measured, or estimated?

   0. Absolutely inaccurately 100. Absolutely accurately

3. How reliable are the instruments used for data collection and measurement?

   0. Extremely unreliable 100. Extremely reliable

4. How accurately are all data transformations done?

   0. Absolutely inaccurately 100. Absolutely accurately

5. How up-to-date are all the data used throughout the entire life cycle of the simulation study?

   0. Absolutely out-of-date 100. Absolutely up-to-date

2.2.9 Model Validation

Substantiating that the experimental model, within its domain of applicability, behaves with satisfactory accuracy consistent with the study objectives is called Model Validation [Balci 1987]. The domain of applicability is the set of prescribed conditions for which the experimental model is tested, compared against the system to the extent possible, and judged suitable for use.

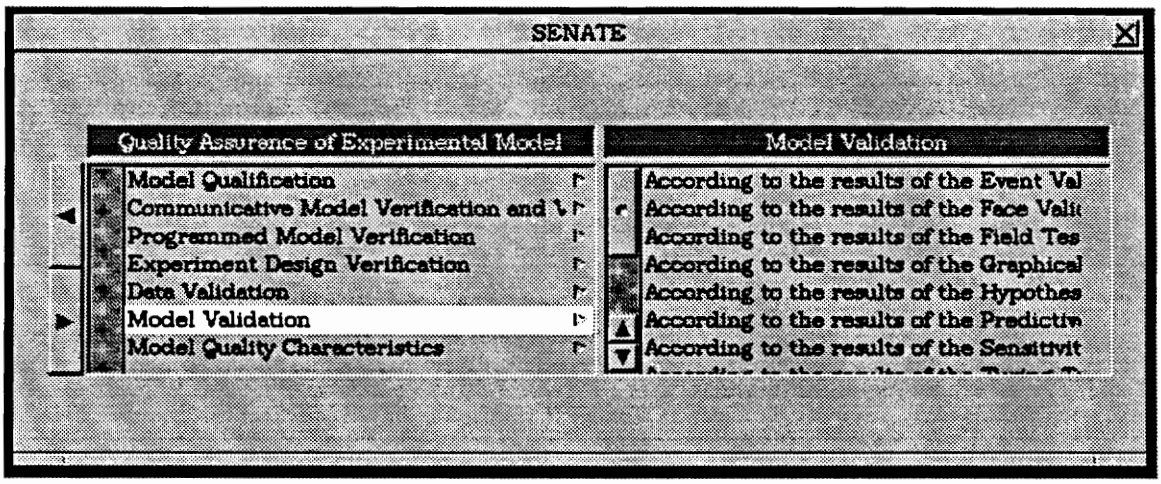


Figure 2.12 Model Validation CAS

The general purpose indicators that we have included in SENATE for this stage of the life cycle are taken from [Balci 1987] and presented below:

1. According to the results of the Event Validation technique (if not used, score zero), how accurately does the experimental model represent the system under study?

0. Absolutely inaccurately 100. Absolutely accurately

2. According to the results of the Face Validation technique (if not used, score zero), how accurately does the experimental model represent the system under study?

0. Absolutely inaccurately 100. Absolutely accurately

3. According to the results of the Field Tests (if not used, score zero), how accurately does the experimental model represent the system under study?

0. Absolutely inaccurately 100. Absolutely accurately

4. According to the results of the Graphical Comparisons (if not used, score zero), how accurately does the experimental model represent the system under study?

0. Absolutely inaccurately 100. Absolutely accurately

5. According to the results of the Hypothesis Validation technique (if not used, score zero), how accurately does the experimental model represent the system under study?

0. Absolutely inaccurately 100. Absolutely accurately

6. According to the results of the Predictive Validation technique (if not used, score zero), how accurately does the experimental model represent the system under study?

0. Absolutely inaccurately 100. Absolutely accurately

7. According to the results of the Sensitivity Analysis technique (if not used, score zero), how accurately does the experimental model represent the system under study?

0. Absolutely inaccurately 100. Absolutely accurately

8. According to the results of the Turing Test (if not used, score zero), how accurately does the experimental model represent the system under study?

0. Absolutely inaccurately 100. Absolutely accurately

9. According to the results of the Confidence Intervals/Regions statistical technique (if not used, score zero), how accurately does the experimental model represent the system under study?

0. Absolutely inaccurately 100. Absolutely accurately

10. According to the results of the Hotelling's T2 Tests (if not used, score zero), how accurately does the experimental model represent the system under study?

0. Absolutely inaccurately 100. Absolutely accurately

11. According to the results of the Goodness-of-fit Tests (if not used, score zero), how accurately does the experimental model represent the system under study?

0. Absolutely inaccurately 100. Absolutely accurately

12. According to the results of the Time Series Analysis statistical technique (if not used, score zero), how accurately does the experimental model represent the system under study?

0. Absolutely inaccurately 100. Absolutely accurately

2.2.10 Model Quality Characteristics

The general purpose indicators that we have included in SENATE for this stage of the life cycle are taken from [Balci 1987] and presented below:

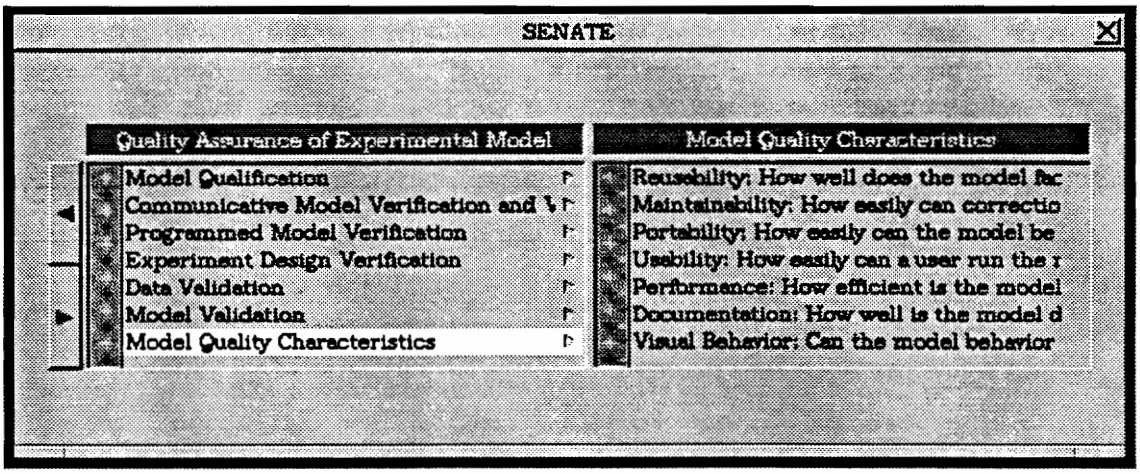


Figure 2.13 Model Quality Characteristics

1. *Reusability:* How well does the model facilitate selective reuse of its components for other purposes (e.g., for the construction of another model)?

   0. No reusability whatsoever 100. Perfect reusability

2. *Maintainability:* How easily can corrections be made in the model to accommodate recognized inadequacies?

   0. No maintainability whatsoever 100. Perfect maintainability

3. *Portability:* How easily can the model be adapted to execute on another computing platform?

   0. No portability whatsoever 100. Perfect portability

4. *Usability:* How easily can a user run the model? How good a user interface does the model have?

   0. No usability whatsoever 100. Perfect usability

5. *Performance:* How efficient is the model execution? Does the model fulfill its

execution objectives without waste of resources?

0. Extremely poor performance 100. Excellent performance

6. *Documentation:* How well is the model documented?

0. Extremely poor documentation 100. Excellent documentation

7. *Visual Behavior:* Can the model behavior be visualized during the course of execution? If yes, how good a visualization does the model provide?

0. No visualization at all 100. Perfect visualization

2.2.11 Quality Assurance of Experimental Model

This CAS includes the Model Qualification, Communicative Miodel Verification, Programmed Model Verification, Experiment Design Verification, Data Validation, Model Validation credibility assessment stages of the simulation life cycle [Balci 1987] [Figure 2.2].

2.2.12 Credibility Assessment of Simulation Results

This CAS includes the Formulated Problem Verification, Feasibility Assessment of Simulation, System Objectives Defination Verification and Quality Assurance of Experimental Model CASs of the simulation life cycle [Balci 1987] [Figure 2.2].

2.2.13 Presentation Verification

The presentation of model results should be verified before they are presented to the decision makers (sponsor) for acceptability assessment. The general purpose indicators that

we have included in SENATE for this stage of the life cycle are taken from [Balci 1987] and presented below:

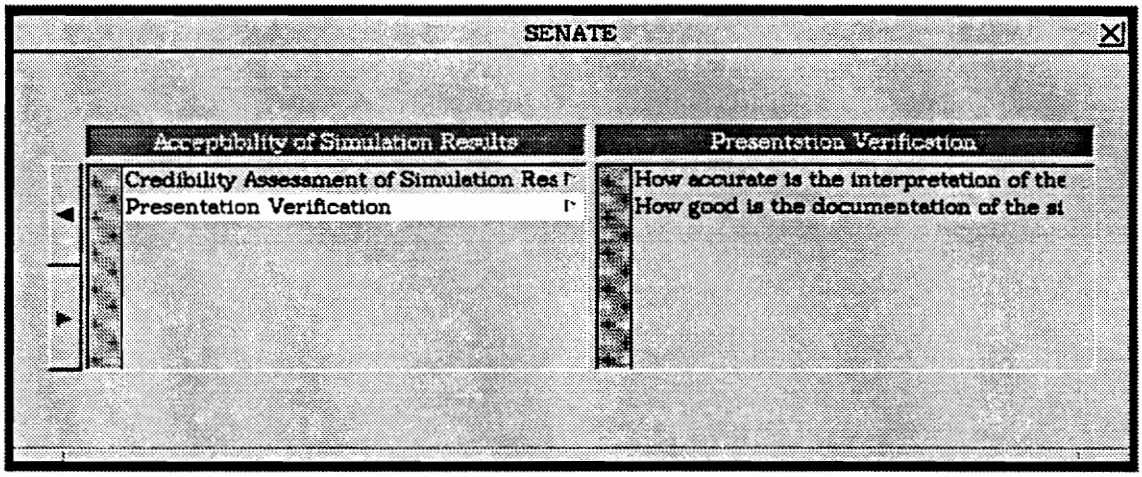


Figure 2.14 Presentation Verification CAS

1. How good is the documentation of the simulation study?

   0. Absolutely bad 100. Absolutely good

2. How accurate is the interpretation of the simulation study results?

   0. Absolutely accurate 100. Absolutely inaccurate

2.4 Simulation Model Development Environment

The Simulation Model Development Environments (SMDE) research project at VPI&SU has been developing an environment that can be characterized as a simulation support environment or a computer-aided simulation engineering environment. The description of the SMDE in this section is taken from [Balci and Nance 1992].The SMDE project has addressed a complex research problem: prototyping a domain-independent discrete-event

simulation support environment to provide a comprehensive and integrated collection of computer based tools to :

- Offer cost effective, integrated and automated support of model development throughout the entire model life cycle;
- Improve the model quality by assisting in the quality assurance of the model;
- Significantly increase the efficiency and productivity of the project team; and
- Substantially decrease the model development time.

The architecture of the SMDE is depicted in the following figure [Figure 2.15] in four layers: (0) Hardware and Operating System, (1) Kernel SMDE, (2) Minimal SMDE, and (3) SMDEs.

- *Layer 0: Hardware and Operating System*

  This layer includes the hardware on which the system is running and the operating system and utilities.

- *Layer 1: Kernel Simulation Model Development Environment*

  This layer basically integrates all SMDE tools into the software environment.

- *Layer 2: Minimal Simulation Model Development Environment*

  This layer provides a comprehensive set of tools which are minimal for the development and execution of a model. Comprehensive implies that the tool set is supportive of all model development phases. Minimal implies that the tool set is basic and general.

- *Layer 3: Simulation Model Development Environment*

  This is the highest level of the environment [as seen from the figure on the next page], expanding on a minimal defined SMDE. In addition to the tool set of the minimal SMDE, it incorporates tools that support specific applications and needed either within a particular object or by an individual modeler. A tool for statistical analysis of output simulation data, a tool for designing simulation experiments, a graphical tool for animation, a tool for input data modeling are examples of tools in layer 3.

  The SENATE system will be included into Layer 3 of SMDE.

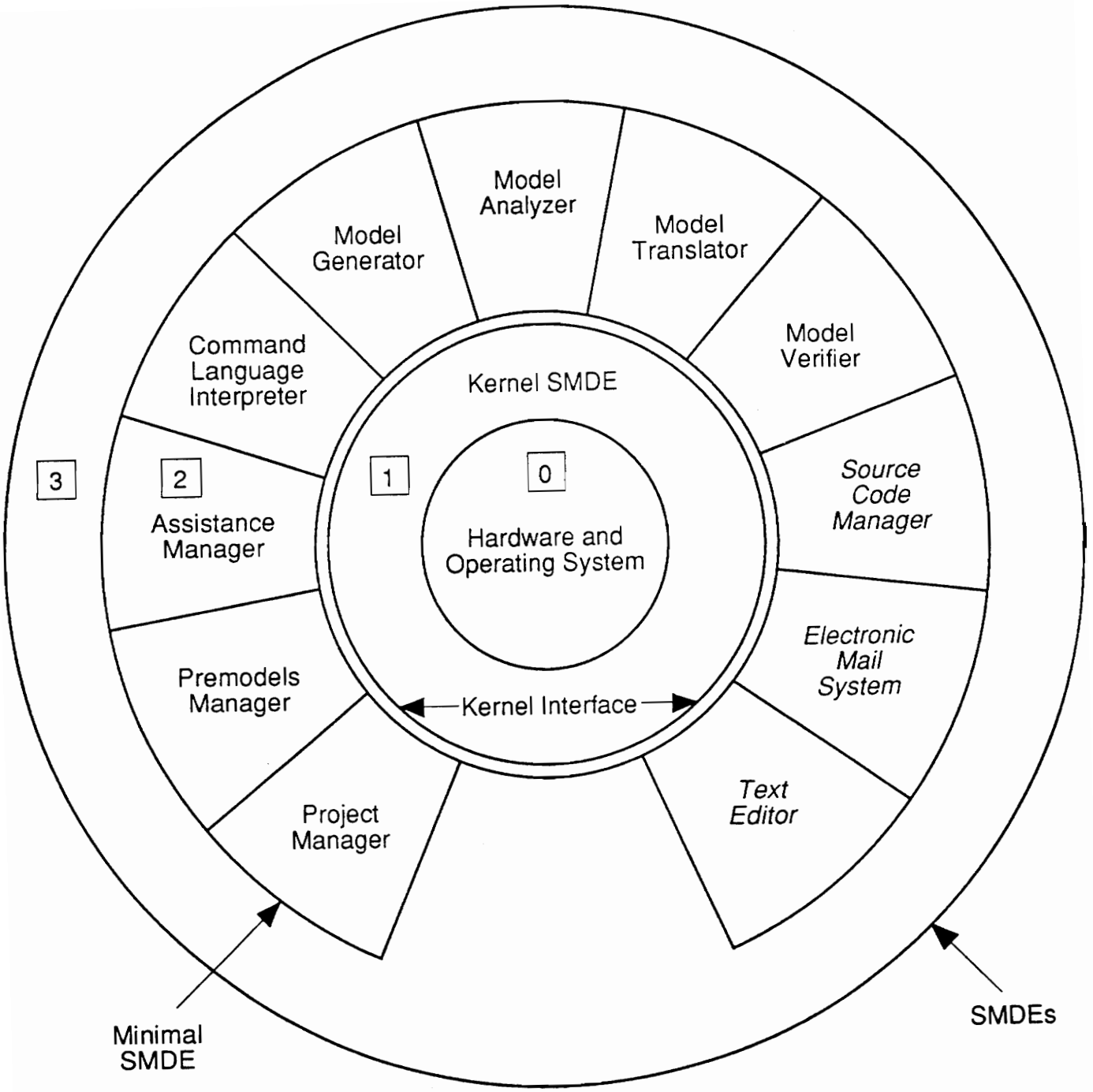


Figure 2.15 SMDE Architecture.

Chapter 3
DESIGN AND IMPLEMENTATION OF SENATE

This chapter presents an overview of the SENATE system development. First, the design principles and objectives are presented. Second, the development environment used to implement the SENATE system is presented. Third, an overview of the implementation of the system user interface is presented. And finally, we explain tips on maintaining the system.

3.1 User Interface Design Philosophy

The objectives of the NeXTSTEP implementation of the system as defined earlier are:

- implement an advanced, modular Graphical User Interface based tool for a system that will help the simulation project management to automate the means of capturing and retaining the expert assessment of the simulation study using indicators;
- the GUI should be easy to use, easy to learn and easy to maintain;
- offer cost-effective, integrated, and automated support for assessing the credibility of simulation results;
- increase the efficiency and productivity of the expert peers;
- substantially decreasing the credibility assessment time; and
- improve the confidence in the assessment by effectively assisting the expert peers during the assessment process.

Hence, the challenge in the design and implementation of the SENATE interface is in providing a powerful tool with an advanced, well-engineered GUI which accommodates the needs of the third party system administrators, the domain experts and the expert peer panel. Therefore, every effort has been put to make the user interface provide easy usability, learnability and maintainability for effective use by all kinds of users.

From the early days of computing, one of the primary objectives of research in

computer science has been the improvement of human computer interaction. To meet this objective while designing the SENATE system we have invested a considerable effort in providing a highly user friendly interface. We have tried to accommodate the following design philosophy and criteria while designing the user interface of the system [NeXT 1992a]:

- The system was designed using the object oriented design and programming paradigm.
- It provides users with task specific information by providing the user with a simple and easy to use direct manipulation user interface and a context sensitive help system.
- The SENATE system allows users to interact with the system using real world metaphors such as buttons, windows, sliders, menus, which relieves the user from memorization and training for a novice user and provides the expert user with the potential to execute a wide range of tasks rapidly. The provision of a powerful sound and hypertext-like context sensitive help system further assists in making the system highly usable and easily learnable.
- The SENATE interface features multi-threaded interaction where a user can initiate multiple tasks independently.
- The NeXTSTEP user interface to the SENATE system was developed in strict adherence to modularity guidelines enforced by object oriented programming. The user interface component and the application component were developed independent of each other. This provided us with the ability to refine the SENATE user interface iteratively without affecting the application component of the system.
- The look and feel of the user interface is consistent within the system and across other NeXTSTEP applications.
- The system provides colorful and intuitive icons in the user interface to make the system attractive, easy to use and easy to learn.
- It uses the mouse as the primary input device (not the keyboard). The mouse being

the more appropriate instrument for a graphical user interface.

- Follow NeXTSTEP user interface standards , conventions and guidelines.
- Use the modal-tool paradigm where the users can change the meaning of subsequent mouse actions by selecting an appropriate tool. For example, double clicking on the SENATE application icon in the workspace manager launches SENATE but once inside the application double clicking on a main browser cell brings up the Functions menu.
- Provide keyboard alternatives for frequently used commands and for commands that are used while working on the keyboard. For example, commands in the Find menu.
- Provide titles that are easy to understand and remember for all the windows, panels, alert panels, icons, browser columns etc.

3.2 Salient Features of SENATE

As described earlier the main objective of this project is to design and implement a cost-effective, well integrated, automated tool that will help the simulation project management to automate the means of capturing and retaining the expert assessment of the simulation study using indicators. Some of the salient features of the system are:

- Both the design and implementation are object-oriented.
- A highly advanced, modular Graphical User Interface (GUI) .
- Facilities for capturing and retaining/storing expert assessment sessions.
- Support for multi-project and multi-user evaluations.
- Highly effective context-sensitive hypertext-like and voice help systems
- Integration with word processors, electronic dictionaries, image and animation display programs, or any other software programs installed on the computer.
- Support for multimedia components such as full-motion video, full-text, sound, animation, TIFF and PostScript images.
- Automatic evaluation report generation facility.

- Indicator string search feature.
- Support for storing information to the SYBASE DBMS.
- Keyboard alternatives for frequently used commands.
- Use of color and intuitive icons in the user interface.
- User and System Administrator Modes of operation.
- Password protection on System Administrator operations.
- Online documentation and user manual.
- Alert panels to assist users.

3.3 Hardware and Software Environment

The SENATE system has been developed on a NeXTSTATION Turbo Color and a NeXTSTATION. The machine combines near photographic quality color and an object-oriented operating and development environment in an easy-to-use, professional color workstation. The NeXTSTATION is a 25-megahertz machine while the NeXTSTATION Turbo Color is a 33 megahertz machine with a Motorola 68040 processor. The 68040 processor is a highly integrated microprocessor providing good computing performance, high data transfer rate, and exceptional reliability. Both machines have 16 MB of memory, built in sound I/O capabilities, 2.88 MB floppy drives and a 400 MB internal hard drive.

The operating system environment used is the NeXTSTEP Release 3. NeXTSTEP is an object-oriented system software that offers an elegant graphical user interface. The NeXTSTEP user interface is based on an advanced, completely object-oriented, development environment using the object-oriented programming (OOP) language. The use of OOP allows developers to produce high quality modular software. In particular the inheritance feature of OOP has allowed SENATE developers to reuse existing code (or objects) provided by the NeXTSTEP Application Kit.

NeXT's operating system is based on the Mach UNIX kernel developed at Carnegie Mellon University, which features shared memory, fast inter-process communication,

multitasking, and network support. NeXT's UNIX is compatible with UNIX 4.3 BSD (Berkely Software Distribution).

3.4 Overview of NeXTSTEP

NeXTSTEP is an interactive windowing environment integrated into an advanced computing environment. For developers, NeXTSTEP is an innovative programming environment that makes it easy to develop advanced object-oriented applications. By providing fundamental building blocks with which to work, NeXTSTEP enhances programmer productivity. By defining the basic features of a consistent user interface, NeXTSTEP makes all applications easy to use. Put simply, the advantage of using NeXTSTEP is that it lets programmers create easy-to-use, leading-edge applications in a fraction of the time required in other windowing environments such as MicroSoft Windows. The features supporting this advantage include:

A graphical development environment that lets you assemble the user interface and other application components in less time than other traditional methods. The features supporting this advantage include [NeXT 1991]:

- Object-oriented programming that modularizes data and procedure, making programs easier to write and maintain.
- A small set of core objects that provide the framework required by any application.
- A rich set of support objects that provide advanced functionality.
- A suite of object-oriented tools to support the application development effort.

3.5 NeXTSTEP Development Tools [NeXT1992d]

In developing the application we have used the following NeXTSTEP development tools:

- Interface Builder- a graphical application for directly manipulating the building

blocks of a program. This enabled us to put greater emphasis on user needs and helped manage the project files.

- Project Builder- a tool for helping developers in creating and maintaining their application projects.
- ApplicationKit- a collection of object-oriented building blocks - this kit provided us with the core framework needed by the system.

3.5.1 Interface Builder

Interface Builder is the central development tool for NeXTSTEP application. From Interface Builder you can add objects to an application, create subclasses, assemble an application's user interface, specify connections between objects for messages, and build the executable file for the application. Interface Builder provides several kinds of windows for controlling its components. In other words, the Interface Builder enables the developer to interactively design and specify the graphical layout and functionality of the graphical user interface for an application development. It also provides the developer with the capability to rapidly prototype the interface and apply iterative refinement of the interface.

The specifications of an assembly of interface objects is saved to a file called an interface file with a *.nib* extension.

The Palettes window offers a variety of graphical objects- including Windows, Controls, Text objects, and Menus- that we have incorporated into our application.

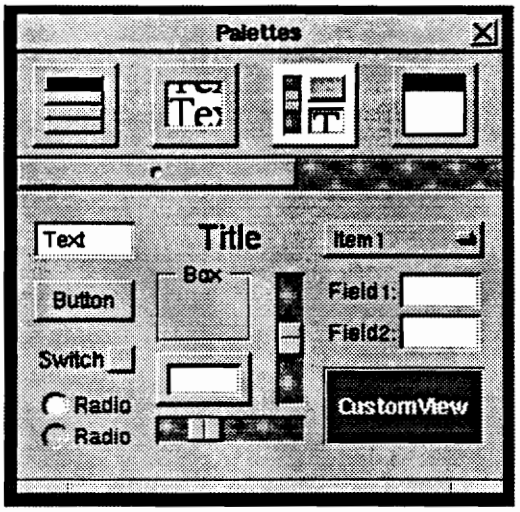
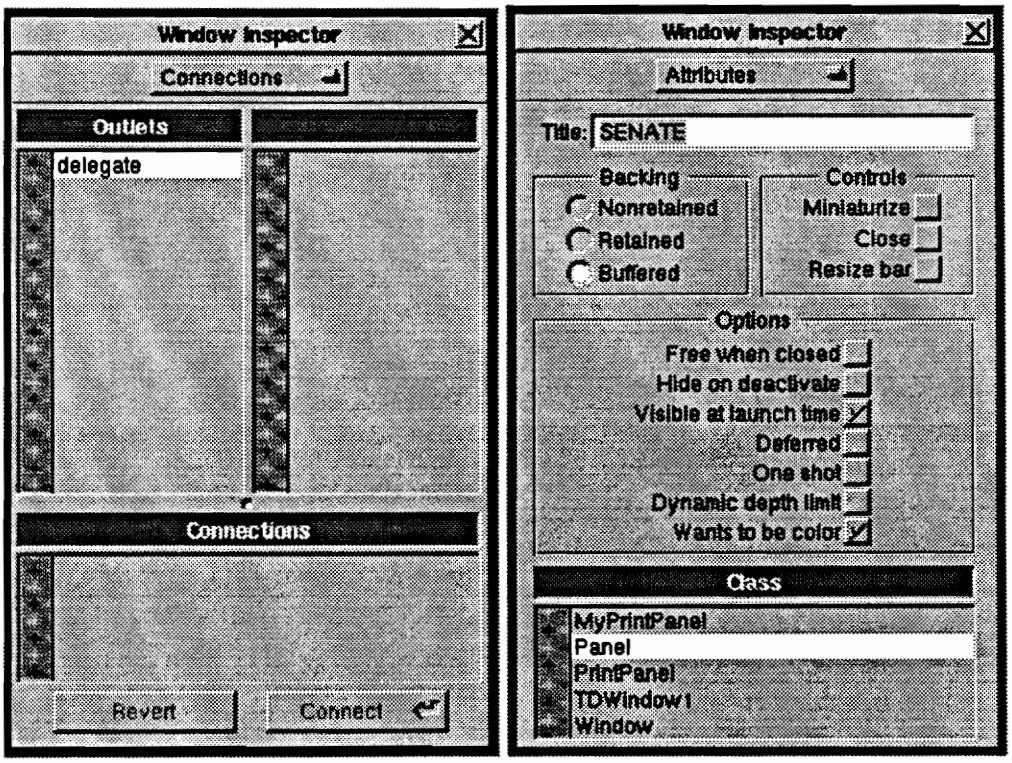


Figure 3.1 Interface Builder Palettes Window

The Inspector window is a multi-purpose tool for controlling elements of the application, including the user interface and the application project.



(i) Connections Inspector (ii) Attributes Inspector

Figure 3.2 Interface Builder Inspector Window

- The Attributes Inspector lets you set the appearance and behavior of user interface objects.
- The Connections Inspector lets you make connections for sending messages between objects.

The Files window provides a top-level view of objects and resources that make up the application. The Classes window lets you browse through the class hierarchy, create custom classes, and add objects of selected classes to the application.

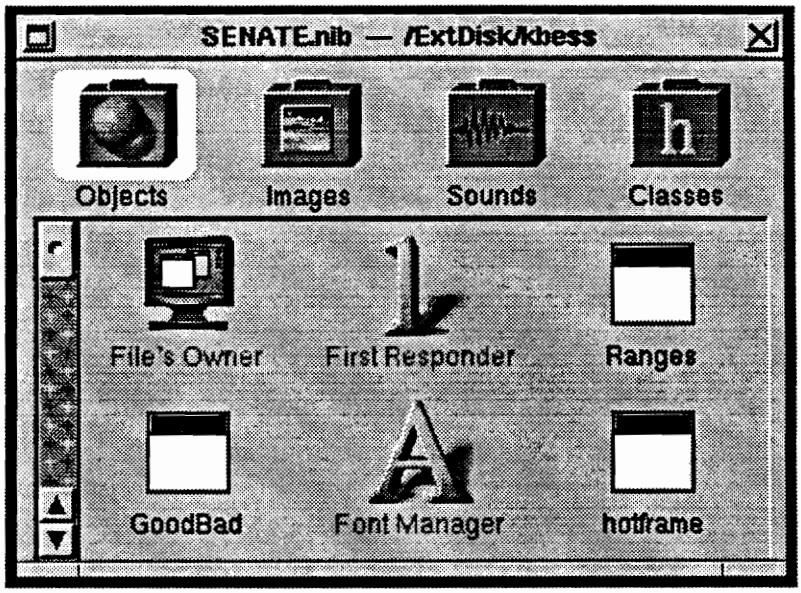


Figure 3.3 Interface Builder Files Window

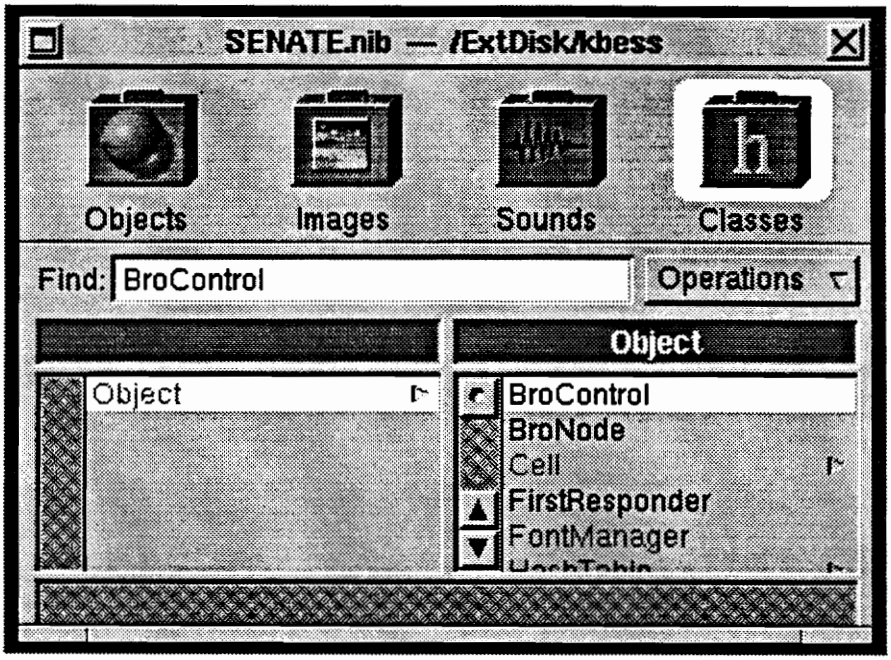


Figure 3.4 Interface Builder Classes Window

3.5.2 Project Builder

Project Builder takes over the application project maintainence role. Project Builder can create a project directory and manage the files used by the project. The files it tracks include the main file, class definition files, icon files, sound files, and others. To keep track of these files, Project Builder creates the file PB.project in the project directory. If you add unique icons for your application and its document files, Project Builder creates and maintains a custom icon header file that establishes the connection between your application and its icons. Project Builder can also know about and manage other component files, such as standard C source code files.

The standard files Project Builder adds to and maintains in the SENATE project directory are:

Category	Description
Classes	Files containing code for custom classes used by an application.
Headers	Files containing declarations of methods and functions used by an application
Other Sources	Files containing code (other than class code) for an application. These may include .m files (containing Objective C code), .c files (containing standard C code), .psw files (containing PostScript code), and other sources. Project Builder automatically adds ApplicationName_main.m to Other Sources.
Interfaces	Files created by Interface Builder for each application and for each new module added to an application.
Images	Files containing images used by an application, including TIFF or EPS files.
Other Resources	Files (such as .snd files) for other resources used by an application.
Libraries	Libraries referenced by an application.

The Project Builder main window helps to maintain, build, and debug the project. Its three modes of operation are:

- Attributes: Set attributes on your project,
- Files: Add, remove, or open project files., and
- Builder: Build the project.

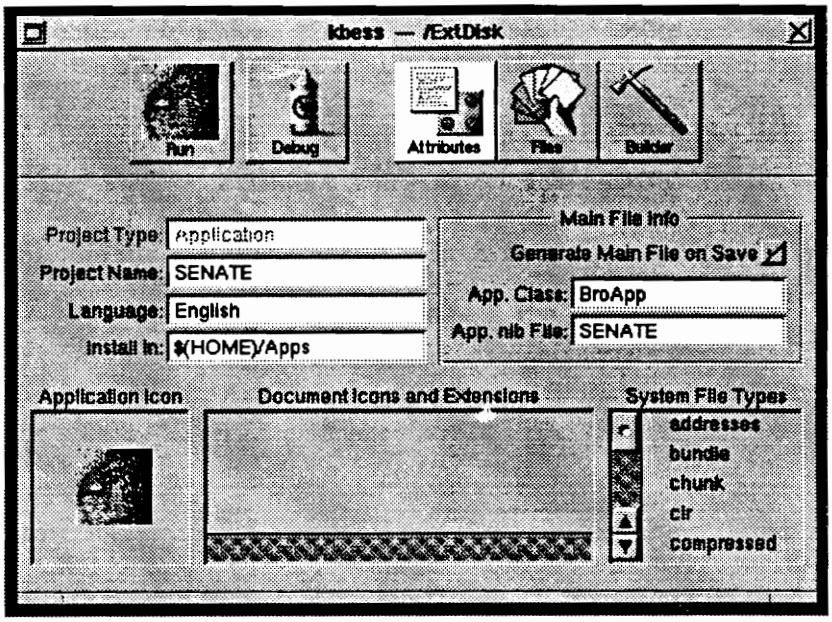


Figure 3.5 Project Builder Attributes Mode Project Window

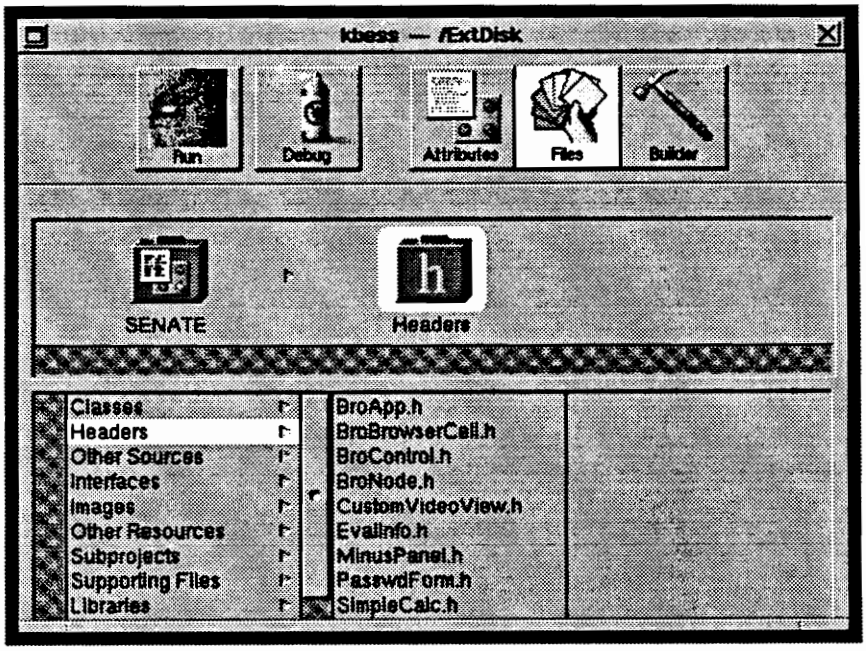


Figure 3.6 Project Builder Files Mode Project Window

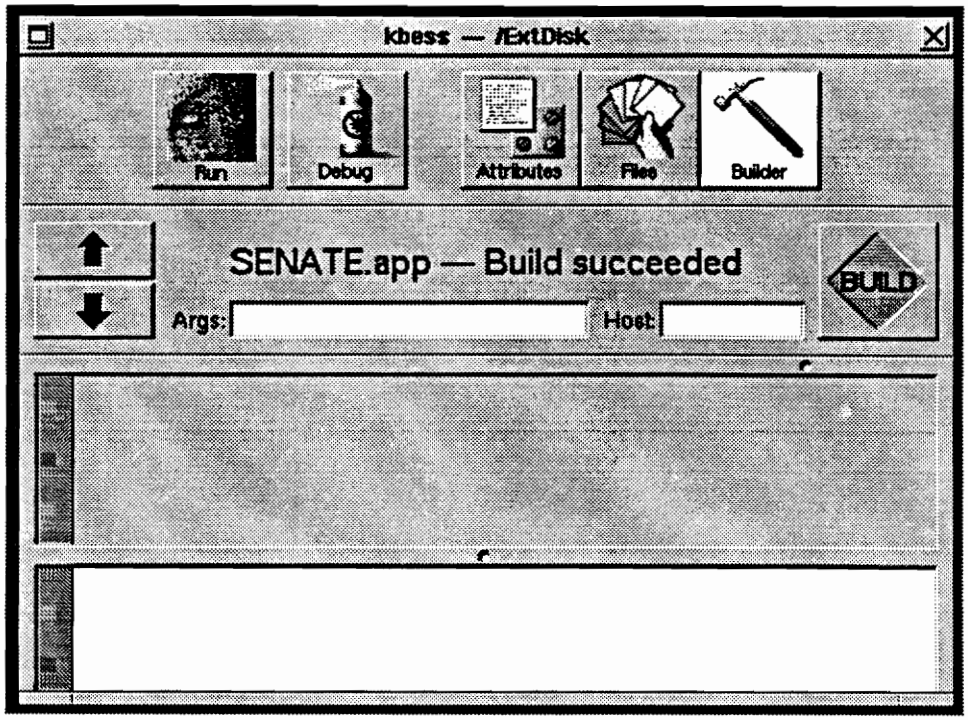


Figure 3.7 Project Builder Build Mode Project Window

3.5.3 Application Kit

The Application Kit defines a set of Objective-C classes and protocols, C functions, and assorted constants and data types that are used by virtually every NeXTSTEP application. The greatest advantage gained by using the Application Kit is it provides the tools for implementing a graphical, event-driven user interface.

- The Application Kit provides ready classes that can be used as ready-made building blocks that can be easily integrated to create complex GUI's .
- The Application Kit makes event-handling extremely simple.

The Application Kit is large; it comprises more than 50 classes and seven protocols. Figure 3.8 shows a map of the Application Kit classes; the classes and protocols.

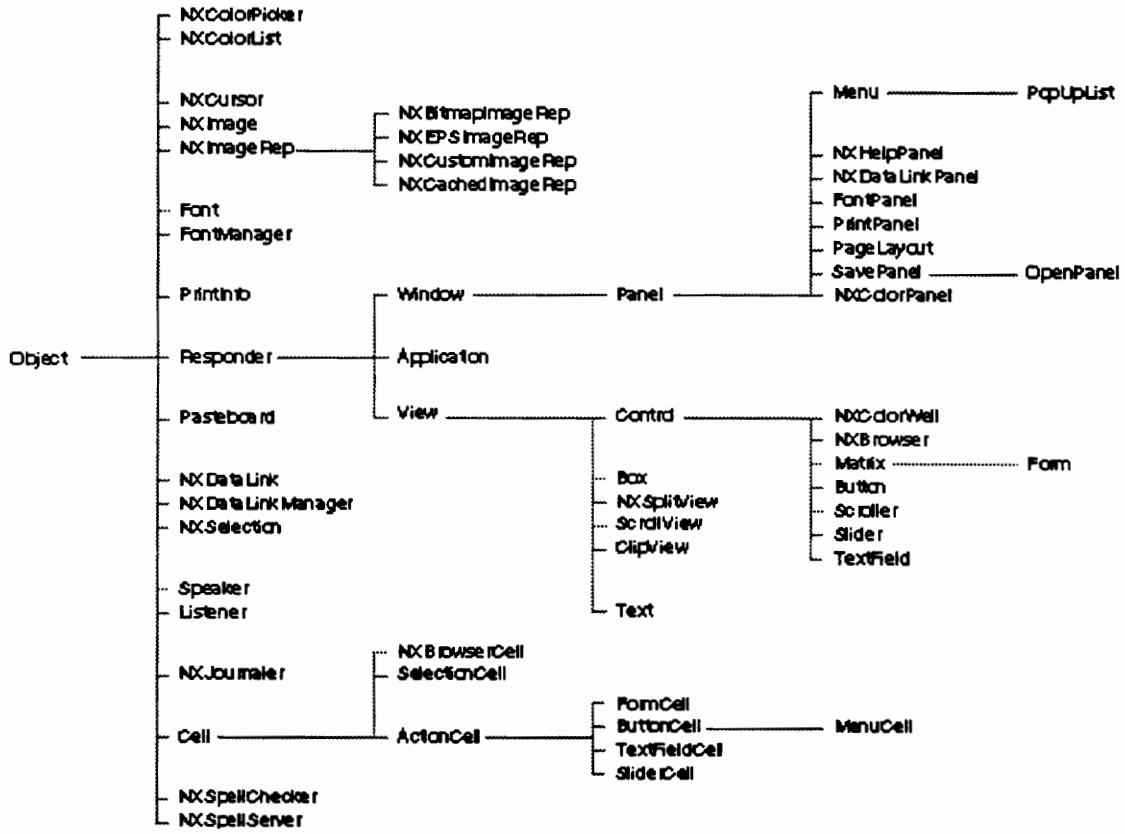


Figure 3.8 Application Kit Classes

3.6 Development Overview [NeXT 1991]

The specific steps in putting together the SENATE system are given below. And they are iterative in nature.

- Assemble the user interface.
- Define the custom classes with the Interface Builder.
- Writing of SENATE specific custom code.
- Connect all application objects.
- Build, debug, rapid-prototype, and run the system.

STEP 1. Design of SENATE

We first defined how the application would work, and specifically how the user will

interact with the system as described in Section 3.1.

STEP 2. Assembling the SENATE User Interface

The user interface development process began by defining the various windows and functionalities required. This initial process was accomplished using the NeXTSTEP Interface Builder where the we assembled the GUI. All the windows, menus, buttons, views, and browsers were created using the Interface Builder. The SENATE project interface files were then saved as .nib files into the four following files:

File	Description
EvalInfo.nib	Interface file for objects used for capturing information on evaluators.
SENATE.nib	Main SENATE interface file; contains most of the interface objects for the application, namely, all the browsers, all the alert panels, all the panels.
SimpleCalc.nib	Interface file for the SENATE calculator
VideoApp.nib	Interface file for SENATE multimedia functionality

The use of multiple interface files improves the performance by not creating interface objects indiscriminately at launch time. But there is a disadvantage too. As a slight performance degradation might be encountered while trying to access an interface object that has not yet been created. In the SENATE interface files organization we try to provide a solution that will let neither of the above mentioned issues considerable affect the performance of the system. The main SENATE.nib file contains all the objects that most users will need while using the system. All those interface objects that will be seldom used are kept in separate interface files, namely, objects for evaluator information, calculator, and multimedia functionality.

The nib file contains [NeXT 1992d]:

- Archived Objects. The Buttons, NXBrowsers, Text Fields, and other objects that you gragged into your application's windows while designing your application's

user-interface are archived in the nib file. The archived file includes the object's class and other attributes, such as its size, location, and position in the view hierarchy.

- Class interface information for any subclasses that you define. At run time, the Application Kit sends mesages to create objects of these classes.
- Information on how outlets can be initialized at run time.
- Information about action messages and their targets.
- Sound and icon data.
- A reference to an owner object. The nib file's owner is an object that's external to the nib file and that is the conduit for messages between the objects that will be unarchived from the nib file at run time and the other objects in the application.

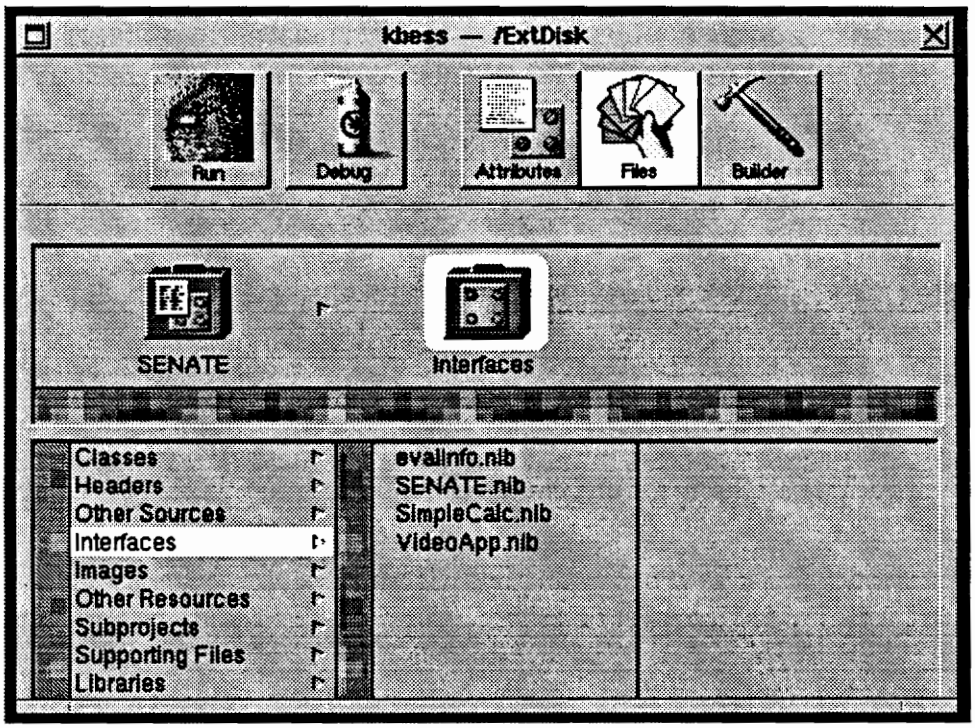


Figure 3.9 SENATE Project Interface Files

STEPS 3 and 4. Developing Custom Objects and Custom Code

The Interface Builder lets us create all the visible objects for the user interface. Now in this step we need to create custom objects that will let us help us manage the SENATE system visible objects and also perform the evaluation information, capture and

storage, and processing. Basically, in this step we implemented all the custom code that was specific to the SENATE system.

The custom classes (or code) was developed using Objective-C. Each class requires two types of files:

- Interface files with .h extensions.
- Implementation files with .m extensions.

The interface files principally contain the description of the objects and the methods that can be invoked by messages sent to the objects belonging to the class. And the implementation file implements those methods. In SENATE all the custom classes are managed by the BroControl class. This class performs a variety of functions such as initialization of the application, coordination between the different classes of the SENATE application, storage of evaluation data.

The following figures list the interface and implementation files in the SENATE project:

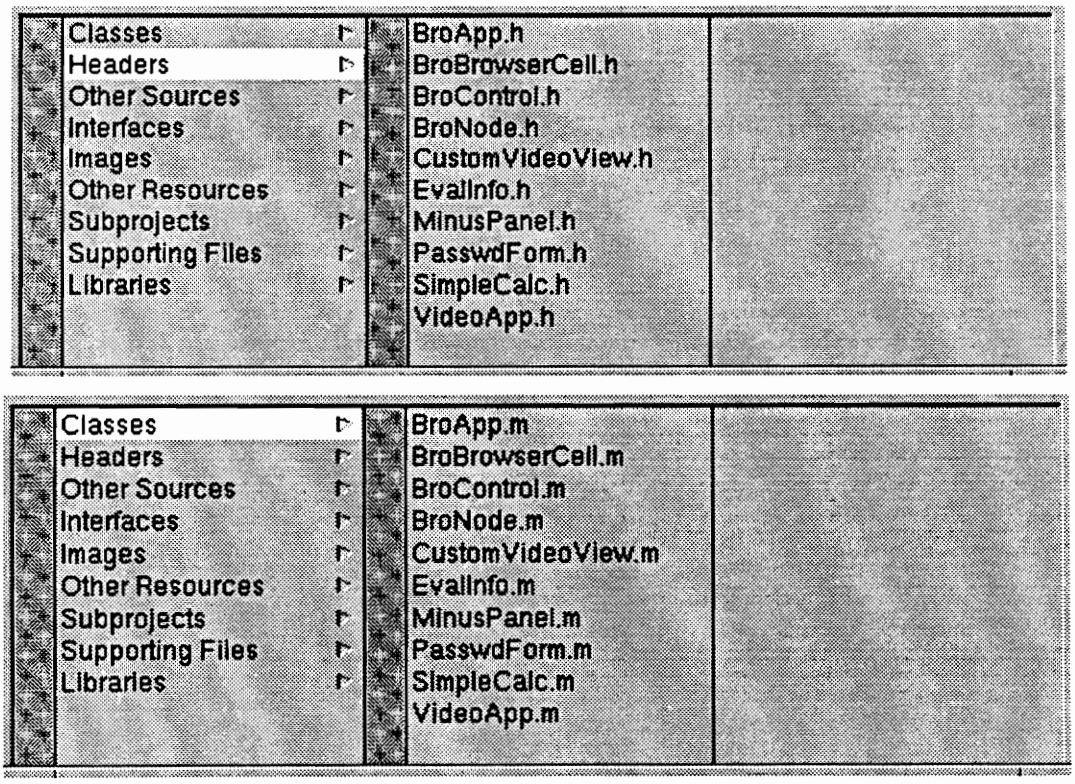


Figure 3.10 SENATE Custom Classes

STEP 5. Connecting all Application Objects

Using the Interface Builder we connected our custom code to the visible interface objects that we built.. See Section 4.5.2.

STEP 6. Build, Debug, Rapid-Prototype, and Run the System

3.7 Main Browser Implementation

3.7.1 Main Browser Design

The SENATE core object in the system is the SENATE Main Browser. We started the system implementation with the browser. The task we were facing was that of developing a hierarchical browser. It should have a series of columns that will be displayed, each containing one or more rows of items. Each row has associated with it a group of items that constitute the next column.

In the Application Kit classes, there is an object *NXBrowser* which provides the functionality we required. However, the NXBrowser only displays data; it does not store any structure. Or in other words only the columns being currently displayed exist. Therefore, to implement the main browser of our system we needed to implement a separate linked list structure that would store all the possible columns simultaneously. Using the Interface Builder we created the browser. The methods- *setBrowser:* and *browser:* in the BroControl class are required to support the NXBrowser that we create using the Interface Builder. Both methods get automatically called by NXBrowser. All the custom classes are managed by the *BroControl* class. This class performs a variety of functions such as initialization of the application, coordination between the different classes of the SENATE application, storage of evaluation data. setBrowser: is called when main browser is created whereas browser: method is called when a main browser column needs loading.

3.7.2 Operations on a Browser Cell

In order to add, delete, modify, score, assign good/bad value, and weigh the current node in the browser we need an underlying structure for each node in the browser. This structure should also have a pointer to its children and parent. To accomplish these tasks in SENATE we have a class of the Application Kit Object class called *BroNode*. The BroNode object in SENATE contains the following information:

- A list of next column data.
- Information about the parent node.
- Weight of the node.
- Score of the node.
- The node string or indicator string.
- Good/Bad Indicator information.

This is the necessary underlying structure. The BroControl *fillMatrix:* method pulls data from this structure and displays it in the NXBrowser object. When a user clicks on any node of the browser that node becomes the current node and is highlighted. The BroControl *resetCurrentNode:* method is responsible for keeping track of the browser current node.

Now in order to keep track of the parent of a node we need to subclass the application kit's *NXBrowserCell*. Its path is Object-Cell-NXBrowserCell. This will allow each cell of the browser to point back at the node that created (parent node) it.

The operations that can be performed on a browser node or cell are:

Operation	**Method**
Create	BroControl *doCreate:*
Delete	BroControl *doDelete:*
Modify	BroControl *doModify:*
Weight	BroControl *doWeight:*
Score	BroControl *doScore:*
Good/Bad	BroControl *doGoodBad:*

3.7.3 Loading and Saving an Indicator Hierarchy

The SENATE system loads up indicators for a project and loads up the set of general purpose indicators if it is a new project. Loading up of indicators is done by the BroControl *restoreIt:* method. Correspondingly, the BroControl *saveIt:* method is responsible for saving the hierarchy of indicators for the current project.

In order to store the indicator hierarchy we require that each node in the structure know how to read and write itself. The standard Application Kit methods for doing this are *read:* and *write:* methods in the controller. The way the class structure is organized these methods need to first call their supper class versions of read: and write: and only then write their instance variables.

The saveIt: method also requires another method that will recursively write all the nodes of the indicators hierarchy into the underlying structure. This method is called *writeColNodes:*. The *setPathString:* method in the BroControl class is responsible for storing the current path in the browser so that when a user comes back for continuing an evaluation session the hierarchy is read back where he left it and the current node in the browser will be at the same place when s/he saved it.

Correspondingly, the restoreIt: method requires supporting methods. Its method *buildRootTree:* reads in the structure that saveIt:'s writeColNodes: method wrote. In addition, it also requires a method that will recursively destroy the current tree if the user had decided to delete a subtree of the hierarchy. This method is called *killAllSubNodes:*. All these methods belong to the controller BroCOntrol class.

3.7.4 Searching for an Indicator

The BroControl method- *searchButtonHit:* is responsible for creating the search panel. The searchButtonHit: method keeps a copy of the search text to determine whether a new search is required or not. When a search is performed a list of all the hits are generated. In the system if the *Next* or *Previous* buttons are clicked and it is not a new

search the system just goes to the next node in the list. Hits also wrap around in both directions.

Each node in the indicator hierarchy is searched with string functions. The *searchList:* method uses the following strategy. Any node that is found to be a hit is remembered with a character string such as 2 3 8 27. The example string represents- column 0 branches at row 2, followed by a branch at row 3 at column 1, then 8 of column 3, and the current node is row 27 of the last column. The list of nodes hit is maintained by the BroControl *foundStor:* method. *SearchList:* goes through the node structure recursively, adding nodes to foundStor: if it makes hits on the way.

Chapter 4
USER'S GUIDE

The SENATE user interface is made up of several different screens that provide the necessary functionality required for assessing the credibility of a simulation study. In this chapter we provide an overview of the SENATE Graphical User Interface (GUI) with the help of a typical user session.

4.1 Launching SENATE

To launch SENATE, select the ~/kbess folder in the File Viewer [Figure 4.1] and double click on SENATE.app.

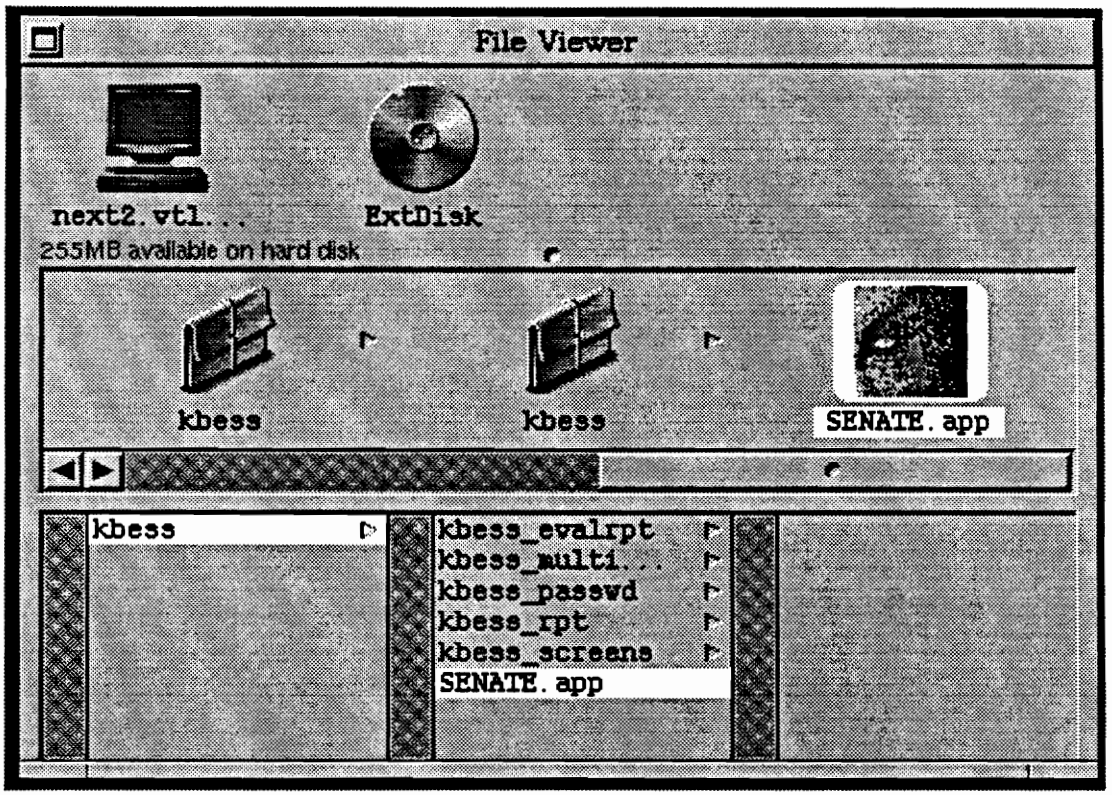


Figure 4.1 Launching the SENATE application

The application is successfully launched when it displays the window shown in Figure 4.2.

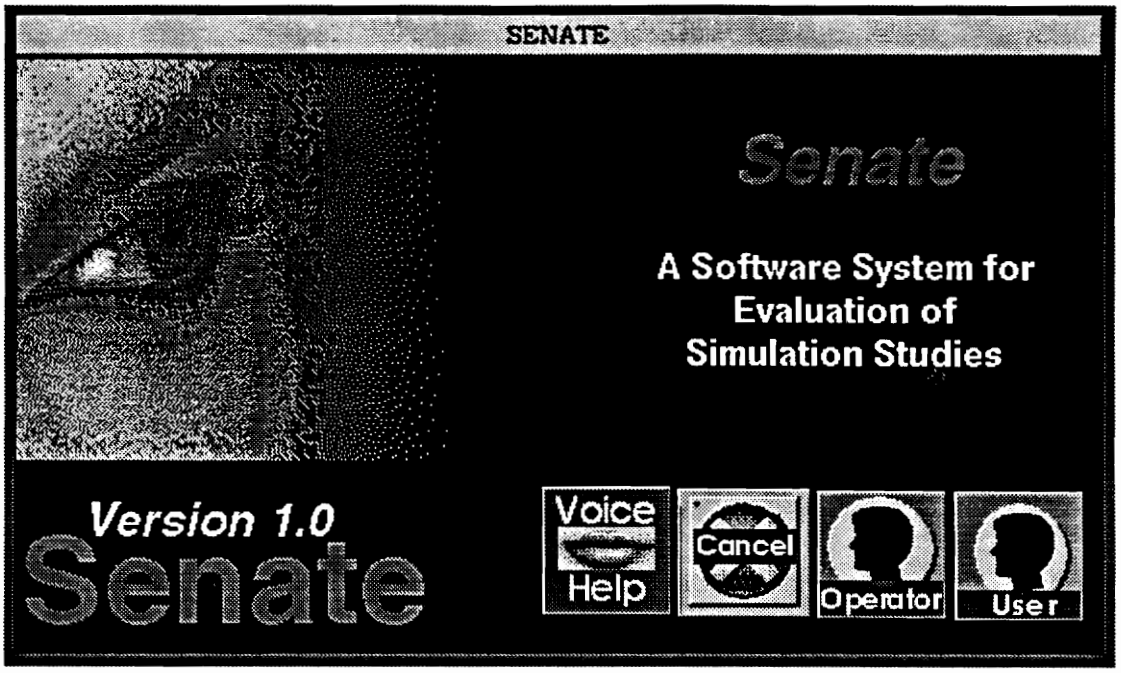
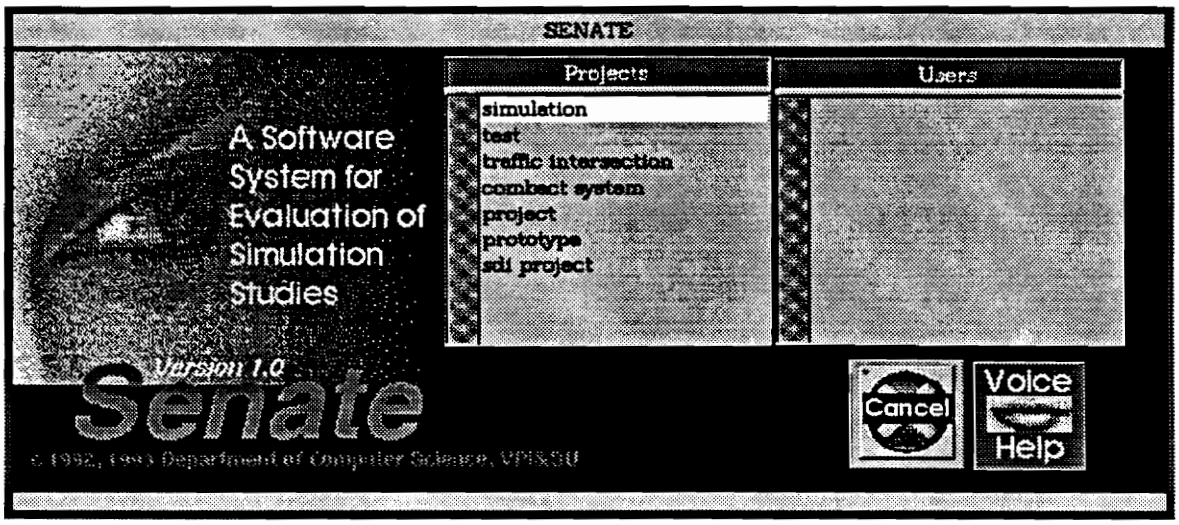


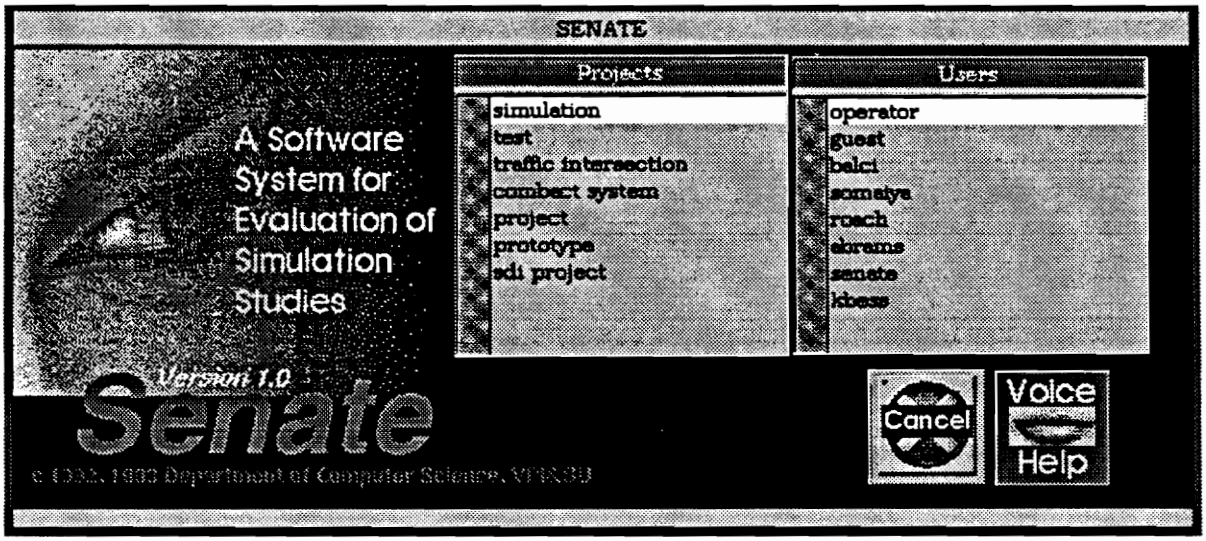
Figure 4.2 Initial Panel

4.2 User Mode

You can now either enter in the Administrator Mode or User Mode by clicking on the Operator or User buttons respectively. Let us first consider a typical user session. Therefore, clicking on the User button will bring up the following panel. In this panel you have two scrollable browsers [Figure 4.3].



(a) Click on a valid project name on the Project Browser and this will load up the User Browser



(b) Click on your user name on the User Browser and the system will ask you for a password

Figure 4.3 Project and User Name Browsers

The browser on the left is the Project browser. It lets you select the name of the project that you want to evaluate by clicking on the browser cell with the correct project

name. And the browser on the right is the User browser. The User browser lets you select your user name. Initially the User browser is blank. It loads up valid users for a particular project, therefore, you first need to select a valid project name before selecting the corresponding user name. Click on the valid project name and then on a user name and this will bring up the User Password Panel [Figure 4.4].

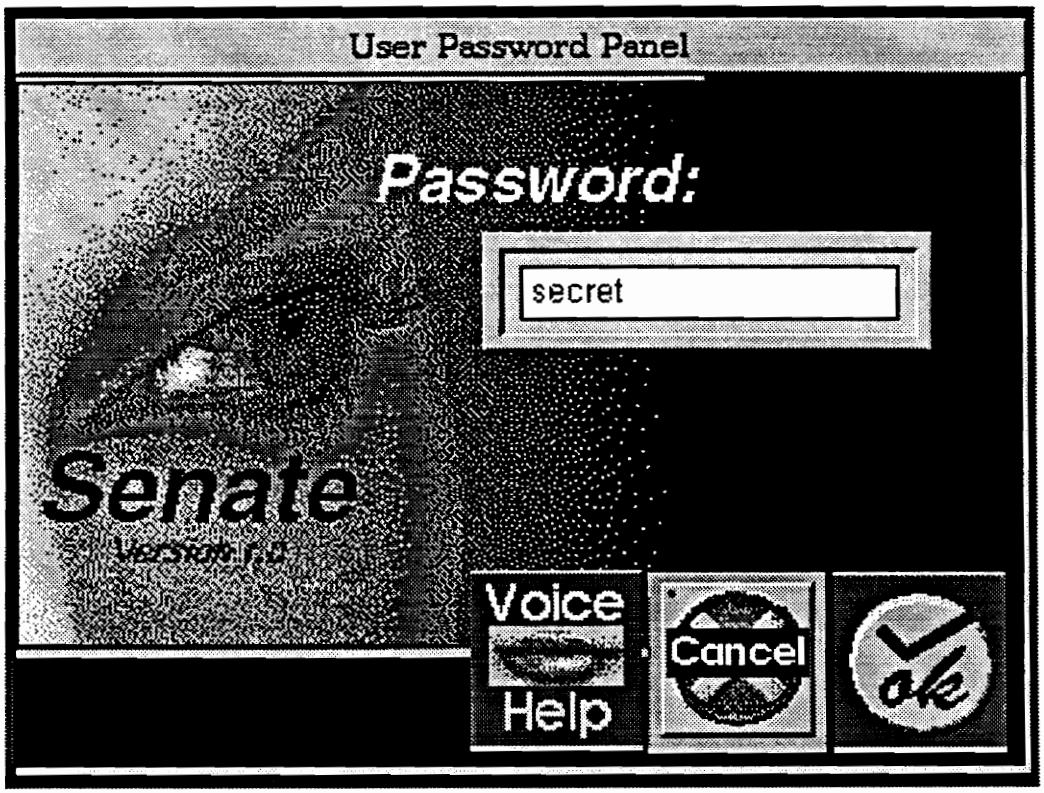
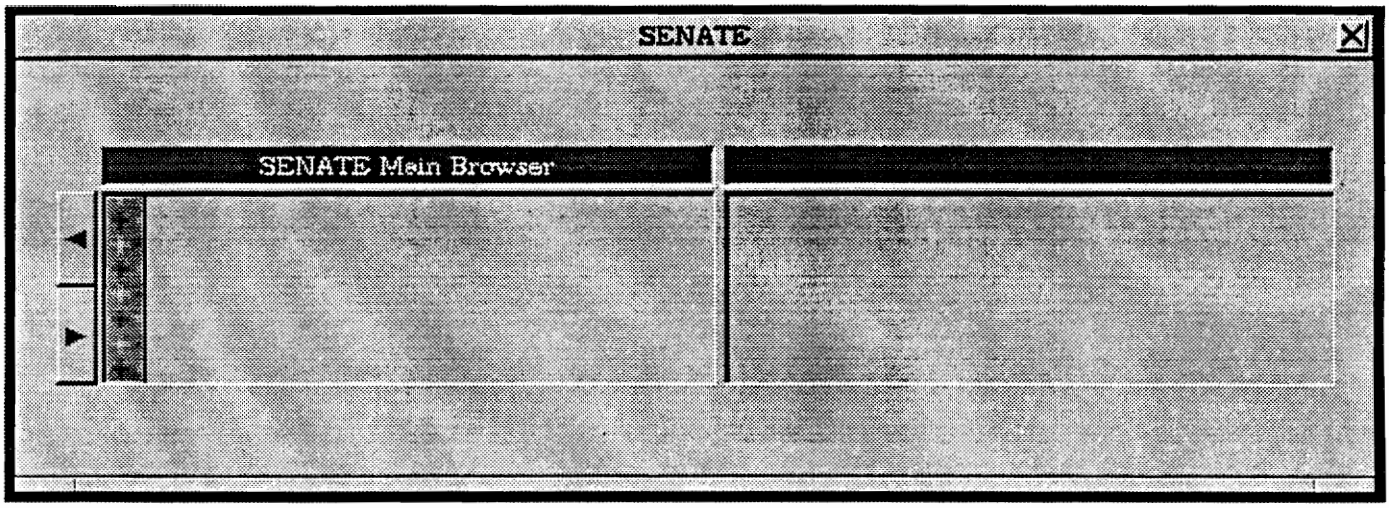
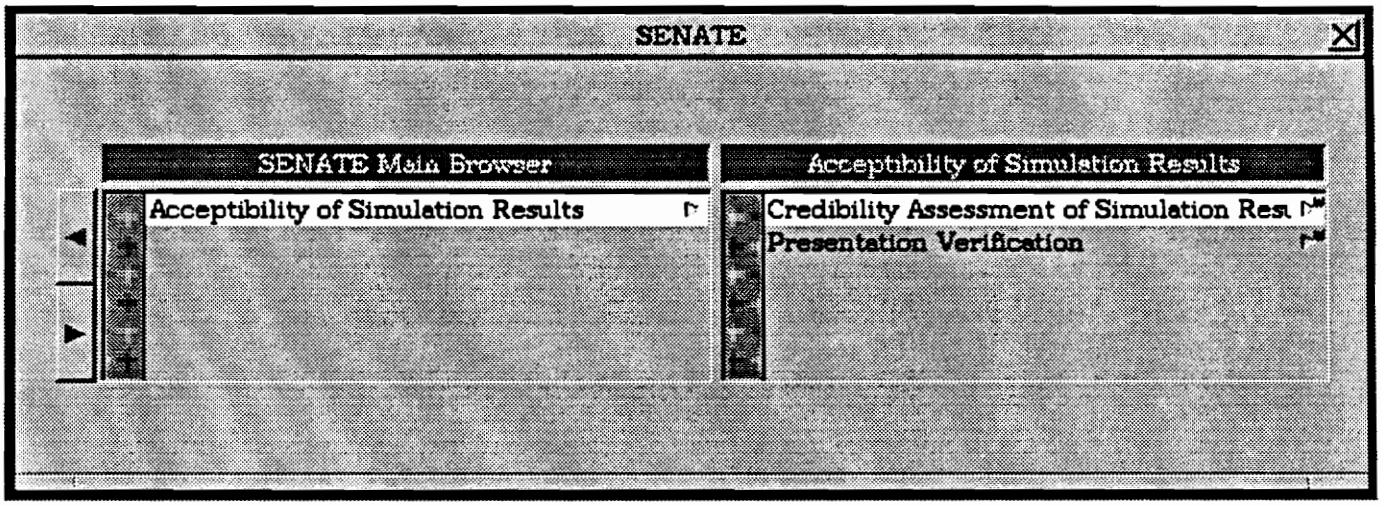


Figure 4.4 User Password Panel

The user can now enter the correct password and click on the "OK" button. If a valid password was not entered by a user the panel will not close after the user clicks on the OK button. If a valid password was entered by a user the User Password Panel will close and the Main Browser will open. Now to load the indicators for the project in the Main Menu click on the Indicators menu cell and then click on Load Indicators menu cell. This will load the indicators associated with the project name and user name that you entered [Figure 4.5].



(a) SENATE Main Browser Window



(b) Click on Indicators-->Load Indicators Menu Command to load the indicators

Figure 4.5 Main Browser

4.3 Main Menu

Menus provide users a point of entry for all the functionality of an application, its obscure and common features alike. SENATE makes use of the menu system's hierarchy to arrange commands in distinct, functionally identifiable menus. A well-defined set of

hierarchical menus are provided which aids users both in finding the commands they need and in understanding the structure of the application.

The menu behaves in a special way:

- All the visible menus for SENATE will disappear when the user starts working in another application. They will reappear when the user returns to the application.
- The SENATE menus are segregated into the front most tiers of on-screen windows. They appear to float above everything else on-screen except attention panels and spring-loaded windows such as pop-up lists.
- Menus can't be miniaturized.
- Menus are hierarchically arranged. Choosing a command in one menu can produce another menu with its own list of commands.

The main menu contains the standard NeXTSTEP menus and commands and the SENATE specific menus and commands. The standard NeXTSTEP commands are similarly in similar menus in other NeXT applications. Many of the standard commands and much of their behavior are supplied by the Application Kit, Project Builder, and Interface Builder. When the SENATE application starts up, by default, the main menu appears in the upper left corner of the screen. Users can change this default location by dragging the main menu to a new position.

The title of the main menu is SENATE which is the name of our application. The various commands in the main menu are shown in Figure 4.6. We have followed all the guidelines that are specified in the NeXT Developer NeXTSTEP User Interface Guidelines Manual.

4.3.1 How SENATE Menus Work

The main purpose of menus is to provide commands for the user to choose. To choose a menu command, the user presses the mouse button as the cursor points anywhere within the current area of the menu and releases it as the cursor points to the

desired command. This can be as simple as clicking the command, or the user can drag through the menu, from command to command. Each command that comes under the cursor while the mouse button is down is highlighted.

Figure 4.6 Main Menu

4.4 The Info Menu

The Info command attaches the Info menu, which contains commands that give general information about the application, as well as let the user set general preferences about how the application works. Info is the first command in the main menu.

The Info Menu contains the commands that let the user get and set information about the application, as a whole. Figure 4.7 shows the submenu under the Info Menu. It has three submenu commands:

a. Info Panel

b. Preferences

The Preferences command brings up the application's Preferences panel, which permits the user to customize the application. At present this command has been disabled.

c. Help

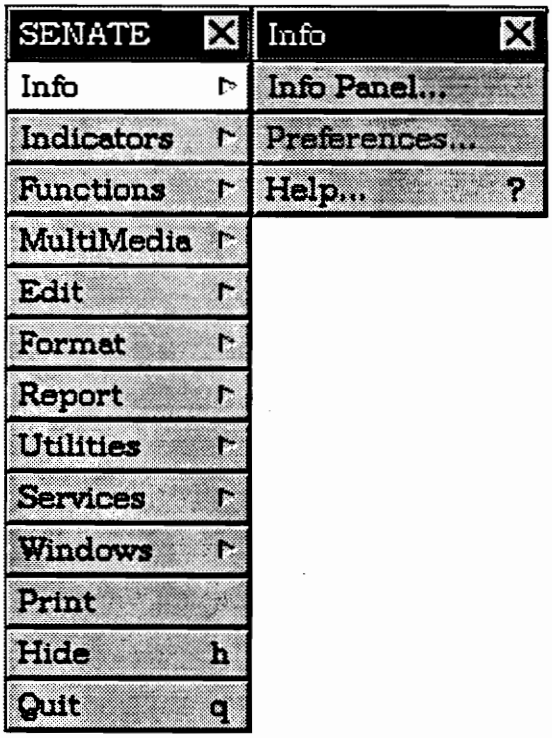


Figure 4.7 Info Menu

4.4.1 Info Panel

The Info Panel command brings up a panel that displays a small amount of basic information about the application [Figure 4.8]. It contains information such as:

- Name of the application.
- The SENATE application icon.
- Copyright information.
- The current version of the application.
- The names of the authors.

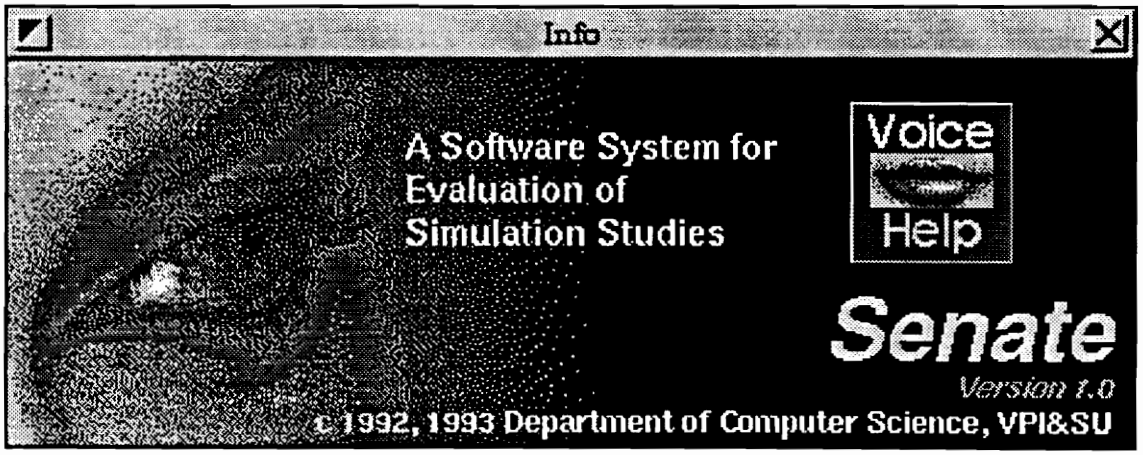


Figure 4.8 Info Panel

4.4.2 Help Command

The Help command brings up a panel with helpful information on how to use the application. SENATE uses the Application Kit Help Panel which is a part of the NeXTSTEP help system. This hyper-text like help system provides users with context sensitive help on the SENATE application and functions. In addition, it also provides help on using the NeXTSTEP user interface. The following section describes the Help System in detail.

4.5 Help System

The SENATE help system has two types of help systems: (a) Voice Help, and (b) NeXTSTEP Help System.

The voice help system works in the following manner. Each window or panel in SENATE has a voice help button. By clicking on it the application plays a context-sensitive voice recording.

The other help system is the NeXTSTEP help system that we have integrated into SENATE. The NXHelpPanel class is the central component of this help system. It provides the Help panel that displays the text and illustrations that constitute SENATE's

help information, and it stores associations of user-interface objects with specific passages of that text.

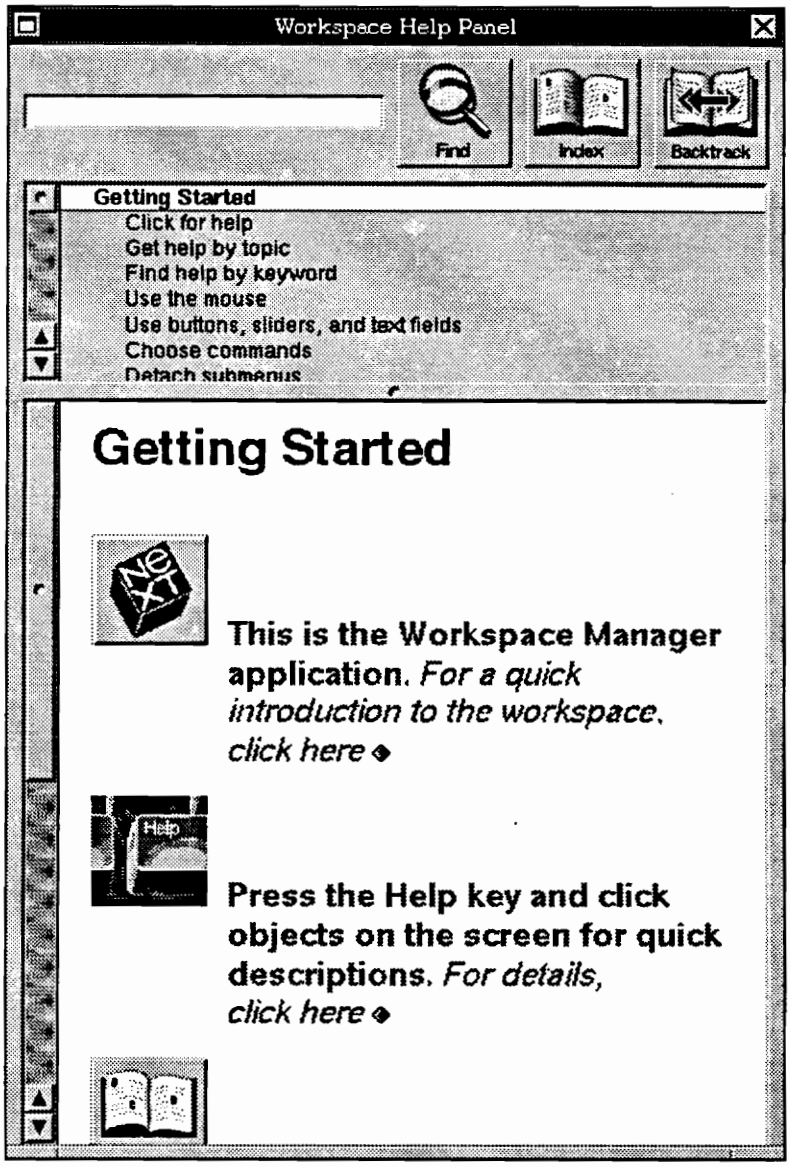


Figure 4.9 System Help system

Users can display the Help panel by choosing the Help command from SENATE's Info menu [Figure 4.9]. The panel employs the metaphor of a book: It displays a table of contents, body text, and an index. Users can browse through the text by clicking entries in the table of contents or index. The panel also supports hypertext-like help links, which

appear as diamond-shaped images within the text and allow the user to easily follow cross references. By using the help cursor and clicking user-interface objects, the user can query the Help panel for information associated with those objects. When the user presses the Help modifier key (or, on older keyboards, simultaneously presses the Control and Alternate keys), a question mark cursor appears. If the user clicks on an object using this cursor, the Help panel displays the associated help text.

4.6 Indicators Menu

The Indicators Menu is shown in Figure 4.10.

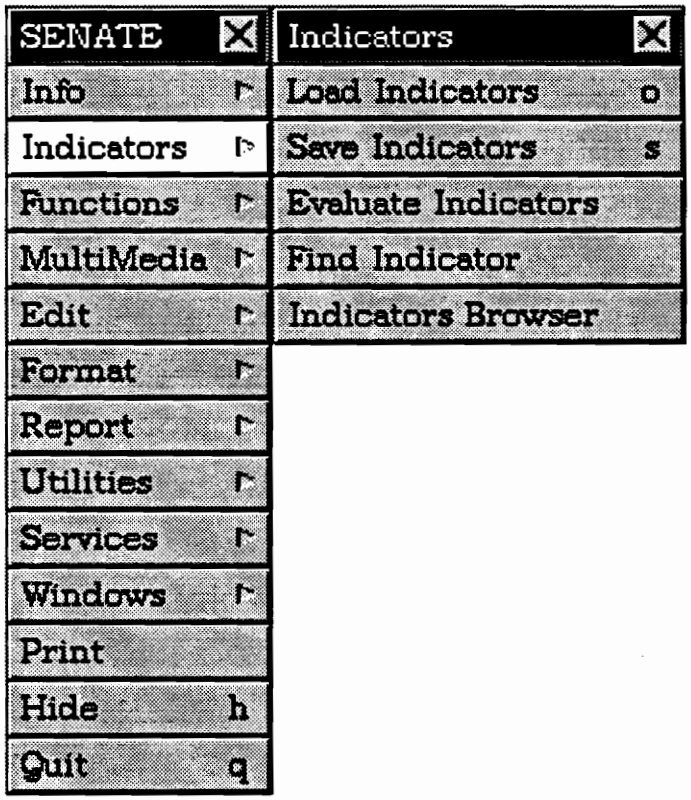


Figure 4.10 Indicators Menu

The Load Indicators command loads up indicators for the project a particular project as described earlier in section 4.2 and Figure 4.5. The Save Indicators command

lets the user save a session. It saves the users current session in to a file ~/kbess_screens/*userName_projectName*. Evaluate Indicators command performs the following operations:

- Checks whether all the indicators have been scores between 0.00 and 100.00. Indicators that have not been scored are assigned a score of 0.00. If any of the indicators have a score which does not lie between 0.00 and 100.00, SENATE will display an alert panel warning the user of the error. It will also highlight the indicator in the Main Indicators Browser.

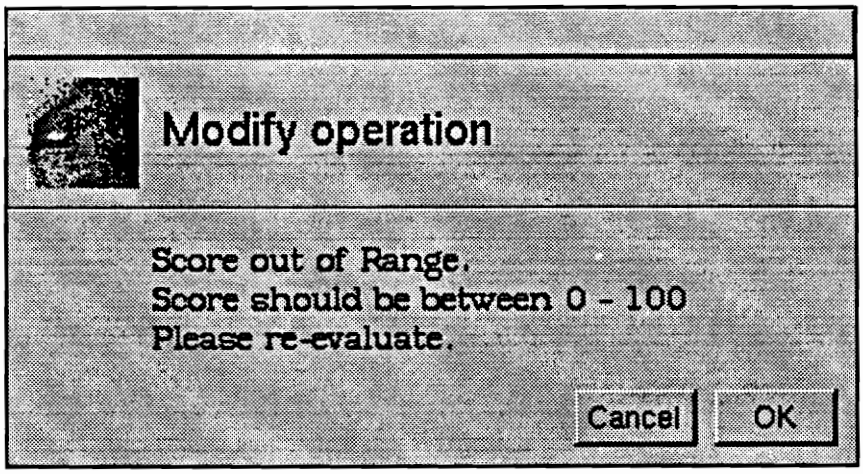


Figure 4.11 Score Alert Panel

- Checks whether all the indicators have a weight between 0.00 and 1.00. Indicators that have not been weighted are assigned a weight of 0.00. If any of the indicators have a weight which does not lie between 0.00 and 1.00, SENATE will display an alert panel warning the user of the error. It will also highlight the indicator in the Main Indicators Browser.

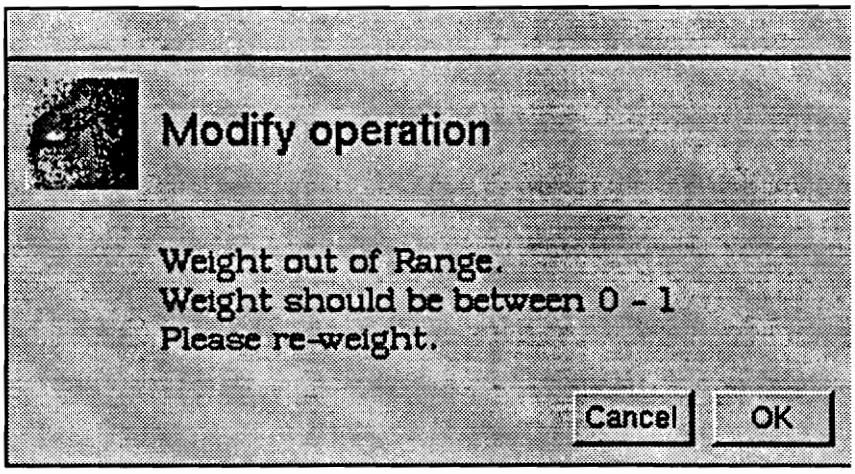


Figure 4.12 Weight Alert Panel

- Checks whether the sum of product of scores and weights of all indicators at a level in the hierarchy add to 100. If at any level they don't add to 100, SENATE will display an alert panel and will also highlight the root indicator of the level.

- Checks whether the sum of weights of all indicators at a level in the hierarchy add to 1. If at any level they don't add to 1.00, SENATE will display an alert panel and will also highlight the root indicator of the level.

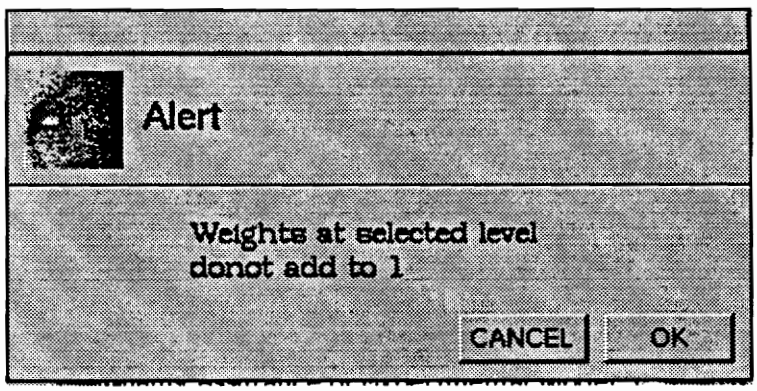


Figure 4.13 Sum of Weights Alert Panel

- Calculates the overall score of an evaluation. The Evaluate Indicators command should be executed only after you have saved your session using the Save Indicators option.

4.7 Functions Menu

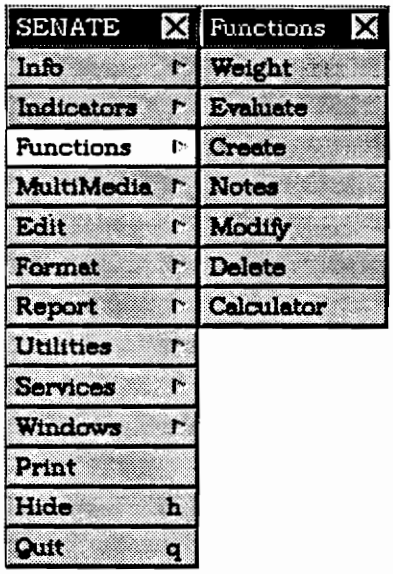


Figure 4.14 Functions Menu

4.7.1 Weight Command

The Weight command lets a user assign weights to the branches in the hierarchy based on his/her experience and training. The user can assign a fractional value between 0 and 1 where 0 represents "not critical at all" and 1 represents "extremely critical". Figure 4.15 shows the panel that comes up when the user clicks on the Weight command.

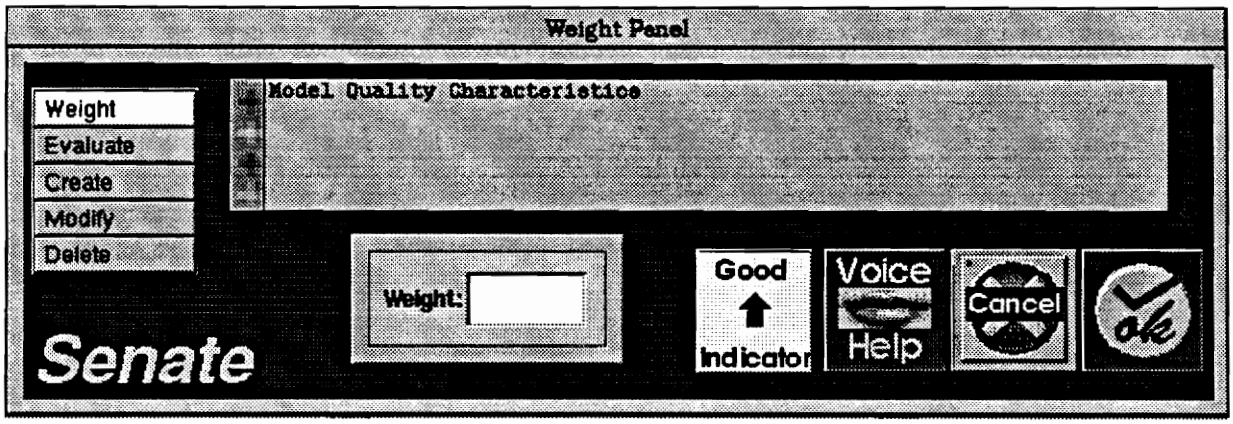


Figure 4.15 Weight Panel

The user assigns a weight for the current indicator in the indicator hierarchy using the Weight Panel shown above. After inputting the weight in the weight form cell, the user needs to click on the OK button. If the weight that the user assigns to an indicator is not an fractional value between 0 and 1 the system will display the alert panel shown in Figure 4.16.

The Good/Bad Indicator button displays whether the current indicator is a Good Indicator or a Bad Indicator. The button is disabled in the Weight Panel, therefore, if the user wishes to change the Good/Bad Indicator flag s/he will have click on the Modify command to do so.

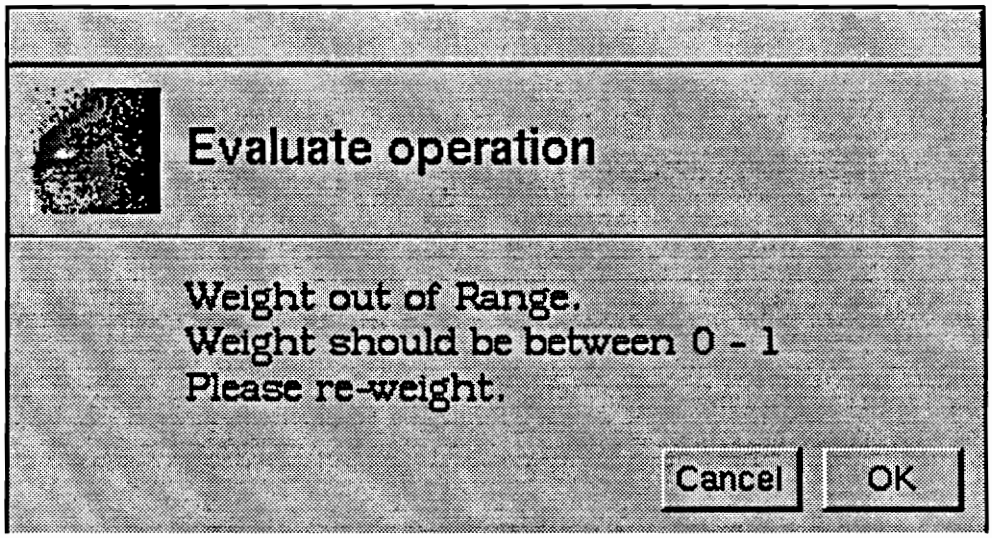


Figure 4.16 Evaluate Operation Weight Alert Panel

4.7.2 Evaluate Command

The Evaluate command lets a user assign a score to an indicator in the hierarchy based on his/her experience and training. The user can assign a fractional value between 0 and 100. The higher the score the more the confidence that the user has on the concept that the indicator represents. Figure 4.17 shows the panel that comes up when the user clicks on the Evaluate command.

The user assigns a score for the current indicator in the indicator hierarchy using the Evaluate Panel. After inputting the score in the score form cell, the user needs to click on the OK button. If the score that the user assigns to an indicator is not an fractional value between 0 and 100 the system will display the alert panel shown in Figure 4.18.

The Good/Bad Indicator button displays whether the current indicator is a Good Indicator or a Bad Indicator. The button is disabled in the Evaluate Panel, therefore, if the user wishes to change the Good/Bad Indicator flag s/he will have click on the Modify command to do so.

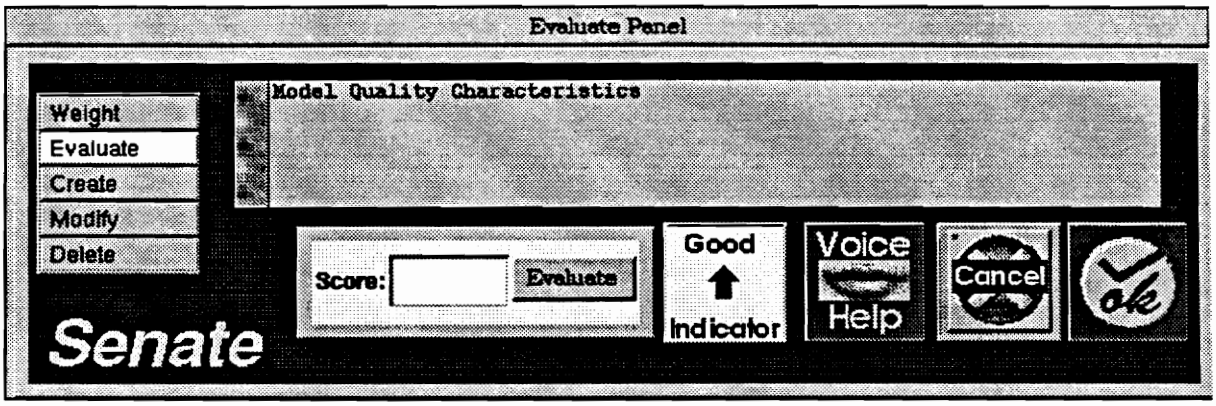


Figure 4.17 Evaluate Panel for Scoring Indicators

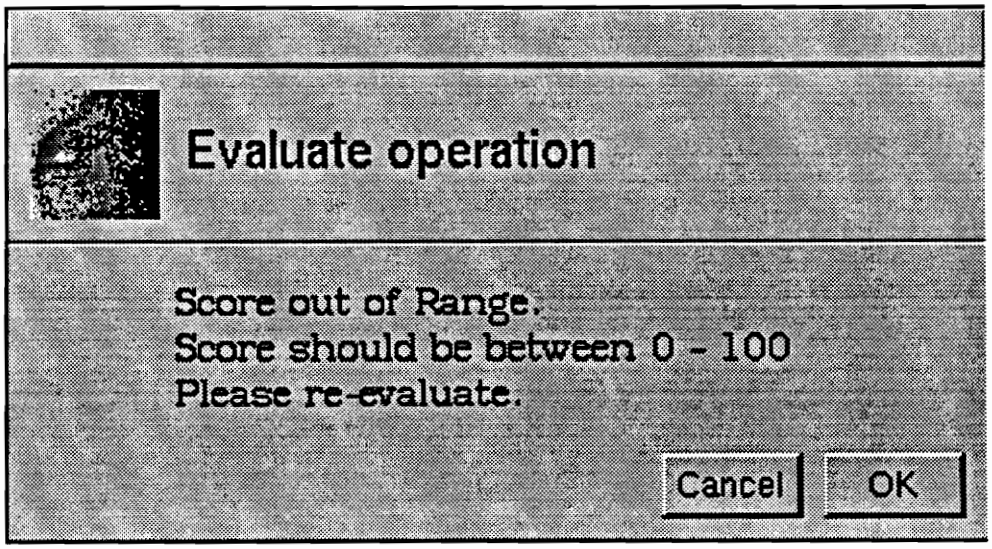


Figure 4.18 Evaluate Operation Score Alert Panel

4.7.3 Create Command

The Create command lets a user create a new indicator in the hierarchy. The user can create a new indicator up to 1024 characters long. Figure 4.19 shows the panel that comes up when the user clicks on the Create command.

The user can create a child indicator for the current indicator in the indicator hierarchy using the Create Panel. After typing the indicator string in the scrollable text view, the user needs to click on the OK button.

The Good/Bad Indicator button displays whether the current indicator is a Good Indicator or a Bad Indicator. The button is enabled in the Create Panel, therefore, if the user wishes to change the Good/Bad Indicator flag s/he will have click on the Good/Bad button to do so. The default is a Good Indicator. The Good/Bad button functions like a toggle switch, therefore, every time the user clicks on it it toggles from Good to Bad and vise versa depending on the current state of the button.

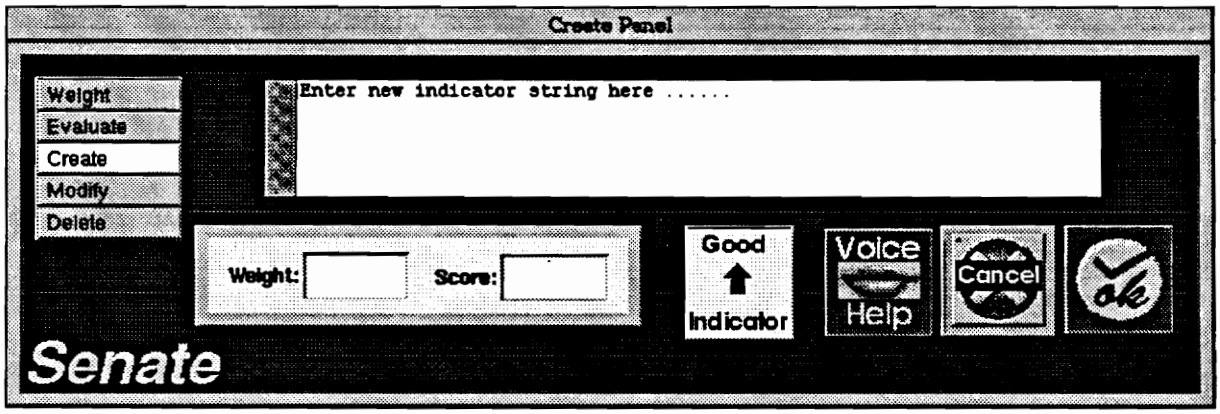


Figure 4.19 Create Panel

The user can also assign the new indicator a weight between 0 and 1 and a score between 0 and 100 using the Create panel weight and score form cells respectively. The Create command is enabled only in the Administrator Mode. The command is disabled in the SENATE user mode system operation.

4.7.4 Notes Command

The Notes command lets a user add an annotation or remark for an indicator in the hierarchy in the user mode. An operator in the administrator mode can use this command to add advise or suggestions for the evaluators. The user can add an annotation or remark for an indicator up to 1024 characters long. Figure 4.20 shows the panel that comes up when the user clicks on the Notes command.

The user adds an note for the current indicator in the indicator hierarchy using the SENATE Notes Panel. After typing the string in the scrollable text view provided, the user needs to click on the OK button.

The Good/Bad Indicator button displays whether the current indicator is a Good Indicator or a Bad Indicator. The button is disabled in the Notes Panel, therefore, if the user wishes to change the Good/Bad Indicator flag s/he will have to click on the Modify command to do so.

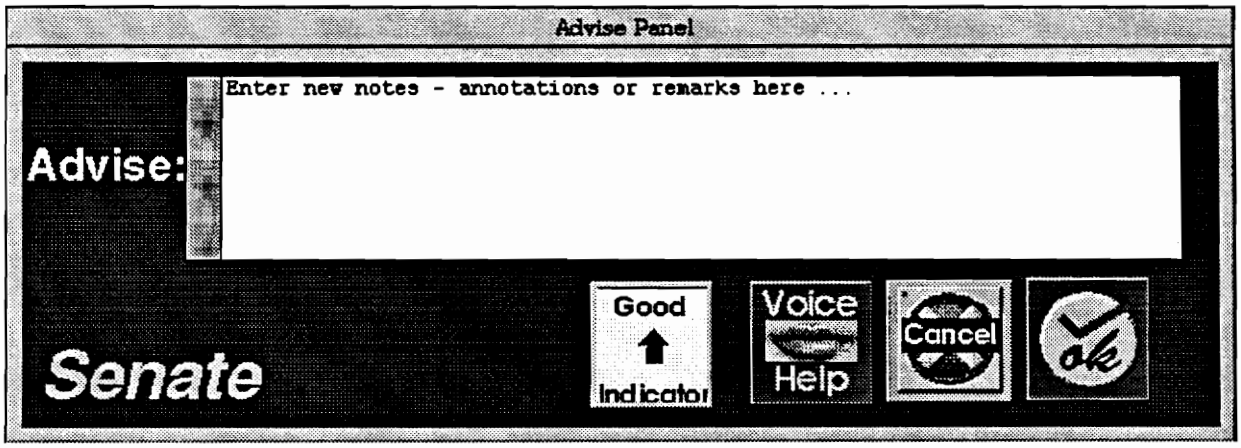


Figure 4.20 Notes Panel

The annotations or remarks that the evaluator adds to an indicator appears in the user's evaluation report. It is highly recommended that users of the system use this feature of SENATE while evaluating the indicators.

4.7.5 Modify Command

The Modify command lets a user modify the properties of an indicator in the hierarchy. The user can modify the indicator string, indicators weight, indicators score, and/or indicators Good/Bad flag. Figure 4.21 shows the panel that comes up when the user clicks on the Modify command.

The user can modify properties of the current indicator in the indicator hierarchy using the Modify Panel. After making the desired modifications, the user needs to click on the OK button.

The Good/Bad Indicator button displays whether the current indicator is a Good Indicator or a Bad Indicator. The button is enabled in the Modify Panel, therefore, if the user wishes to change the Good/Bad Indicator flag s/he will have click on the Good/Bad button to do so. The Good/Bad button functions like a toggle switch, therefore, every time the user clicks on it it toggles from Good to Bad and vise versa depending on the current state of the flag/button.

The Modify command lets the user modify all the properties associated with an indicator except the notes string. To modify the notes string the user has use the Notes command.

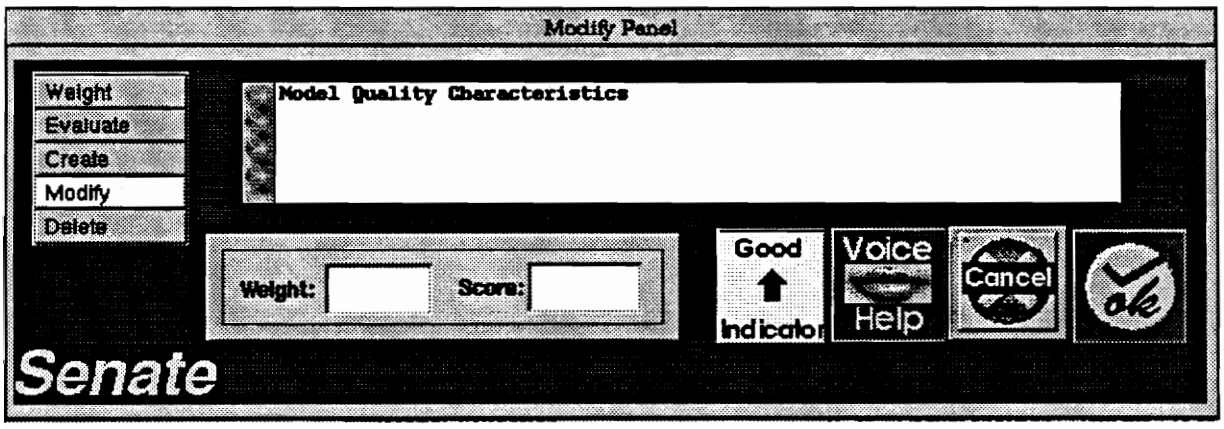


Figure 4.21 Modify Panel

4.7.6 Delete Command

The Delete command lets the user delete an indicator in the hierarchy. The user can delete the indicator string, indicators weight, indicators score, indicator notes, and indicators Good/Bad flag values associated with an indicator using this command. Figure 4.22 shows the panel that comes up when the user clicks on the Delete command.

The user can delete properties of the current indicator in the indicator hierarchy using the Delete Panel. The user needs to click on the OK button to confirm the operation. S/he may choose to cancel the operation by clicking on the Cancel button. If an indicator is not a leaf node then the system will display an alert message as shown in Figure 4.23.

The Delete command is enabled only in the Administrator mode. In the User mode this command remains disabled through out an evaluation session. Or in other words only an operator has the permission to delete an indicator or a hierarchy of indicators.

Figure 4.22 Delete Panel

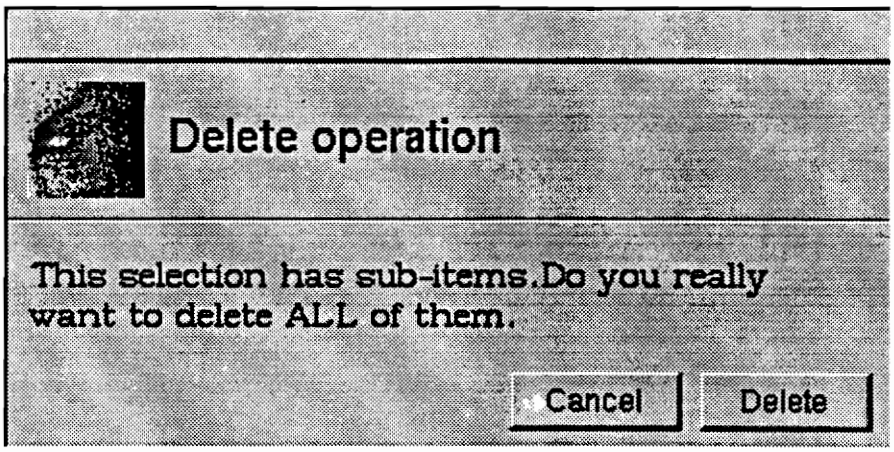


Figure 4.23 Delete Alert Panel

4.7.7 Calculator Command

The Calculator command brings up a simple calculator as shown in Figure 4.24 below. The calculator tool has been implemented to help the user perform arithmetic calculations during an evaluation. It is a very helpful tool for calculating indicator scores and weights.

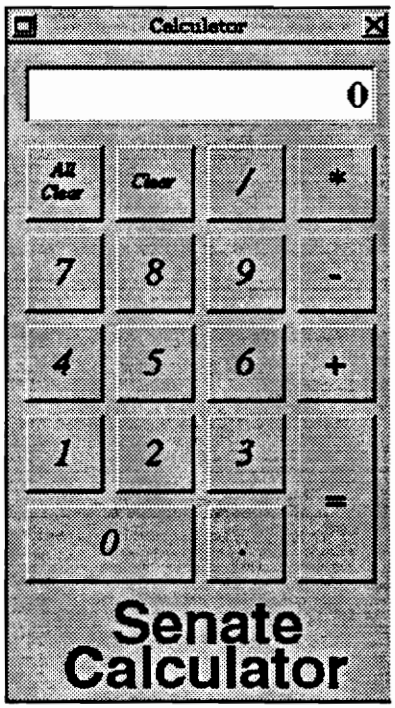


Figure 4.24 Calculator Tool

4.8 Multimedia Menu

The Multimedia Menu is shown below:

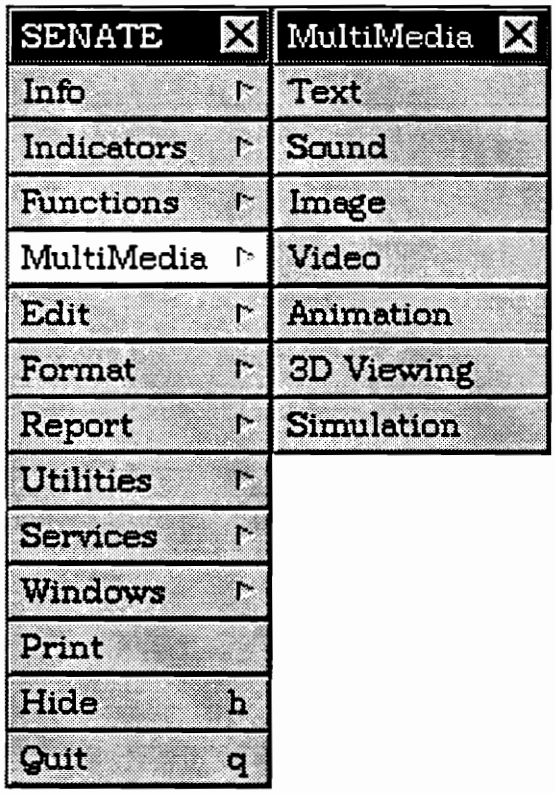
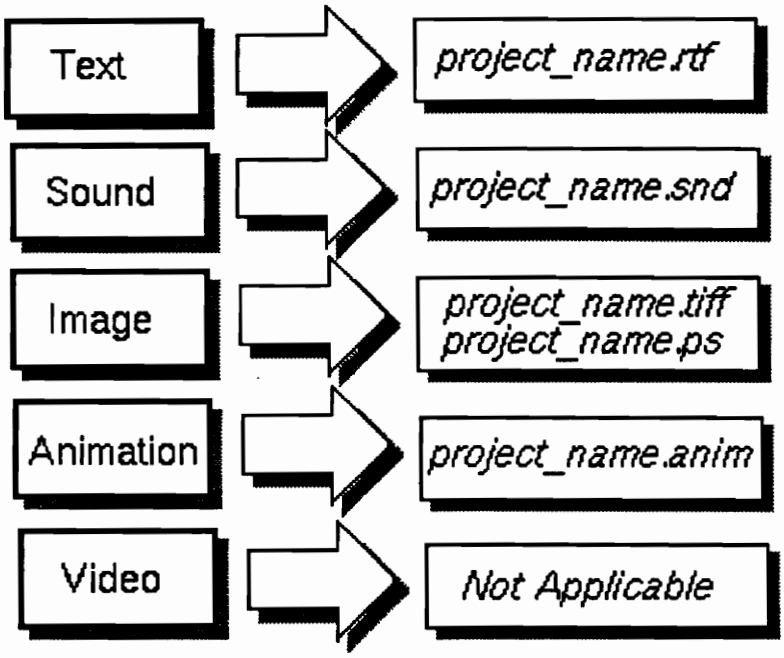


Figure 4.25 Multimedia Menu

This feature lets the SENATE system integrate multimedia information to help the users understand and/or visualize the simulation study that they are evaluating. This feature integrates text, image, sound, animation and full motion video information with each project under the system. The operator needs to place the multimedia information in the ~/kbess_multimedia directory. Information gets automatically associated with a particular project depending on the file name that the operator places it into. The following table describes the file naming conventions to use while associating different kinds of multimedia information to a project [Table 4.1].

Table 4.1 Multimedia File Naming Conventions [all files should be placed in ~/kbess_multimedia directory]



When a user clicks on the Text command, the ~/kbess_multimedia/project_name.rtf file is displayed to the user. This file may contain any textual information that could aid the user in assessing the credibility of the simulation study results. For example, an operator can place the simulation study project report in this file. The file is displayed using the /NextApps/Edit.app application. Once the user has studied the information s/he may quit the Edit application to return to the SENATE application.

When a user clicks on the Sound command, ~/kbess_multimedia/project_name.snd file is displayed to the user. This file may contain any audio recording that could aid the user in assessing the credibility of the simulation study results. For example, an operator can place interviews with the experts who conducted the simulation study in this file. The recording is played using the /NextDeveloper/Demos/Sound.app application. Once the user has heard the information s/he may quit the Sound application to return to the SENATE application.

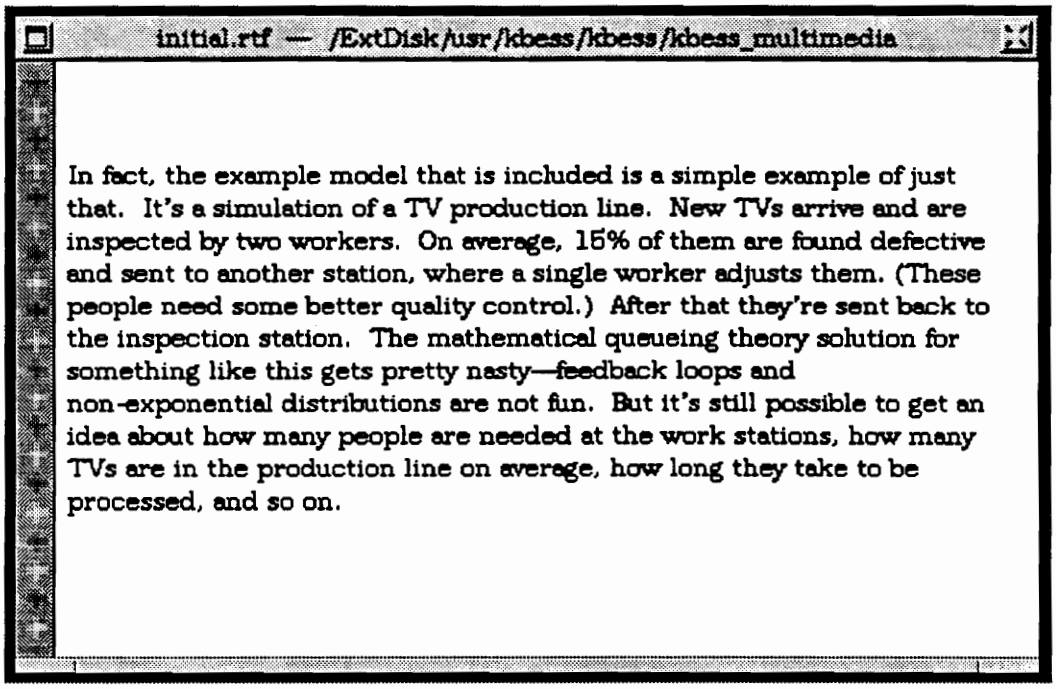


Figure 4.26 Text Command: Edit.app Window

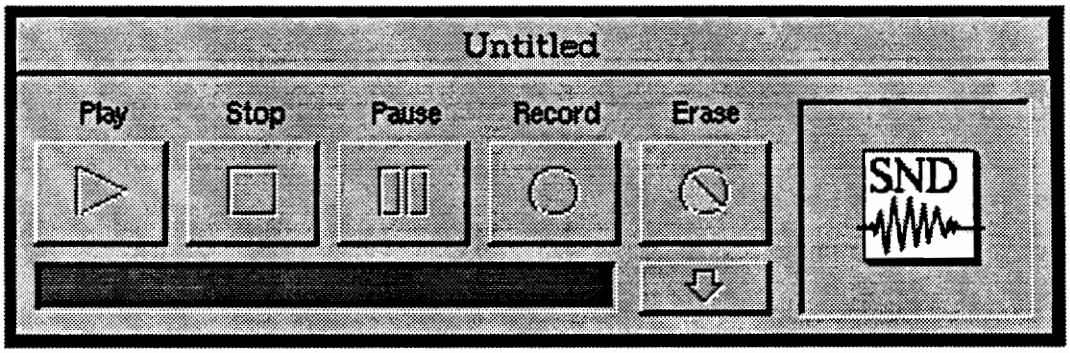


Figure 4.27 Sound Command: Sound.app Panel

Clicking on the Image command: the ~/kbess_multimedia/project_name.ps or .tiff or .eps file is displayed to the user. This file may contain any image that could aid the user in assessing the credibility of the simulation study results. The image is displayed using the /NextApps/Preview.app application. Once the user has viewed the information s/he may quit the Preview application to return to SENATE.

Clicking on the Animation command ~/kbess_multimedia/project_name.anim file executed. This file may contain any animation that could aid the user in assessing the credibility of the simulation study results. The SENATE interface for displaying animations is basically a TIFF sequence animator. The standard file structure for an animation is a directory ~/kbess_multimedia/project_name.anim with the contents project_name1.tiff, project_name.2.tiff, and so on. Where project_name is the name of the animation. The base name of the directory (less '.anim') must be the same as the names of the TIFF files. Each TIFF file should contain one TIFF image. The TIFF files must be numbered consecutively starting from 1. The program is smart enough to figure out how many frames there are in the animation.

The Video command can be used to display full motion video using the SENATE Video interface. This feature is possible only on NeXT machines with a NeXTDimension board. This function has been implemented the using NXLiveVideoView class. In addition, to displaying full motion video it also includes image grab and video output of graphics. See Figure 4.28.

The last option in the Multimedia menu is the Ents application [Figure 4.29]. Ents is a program for doing discrete event simulation. Simple to moderately sized systems such as complex queuing systems or manufacturing problems can be modeled by relative novices; the output can be used to suggest better systems or decide among several possible alternative systems.

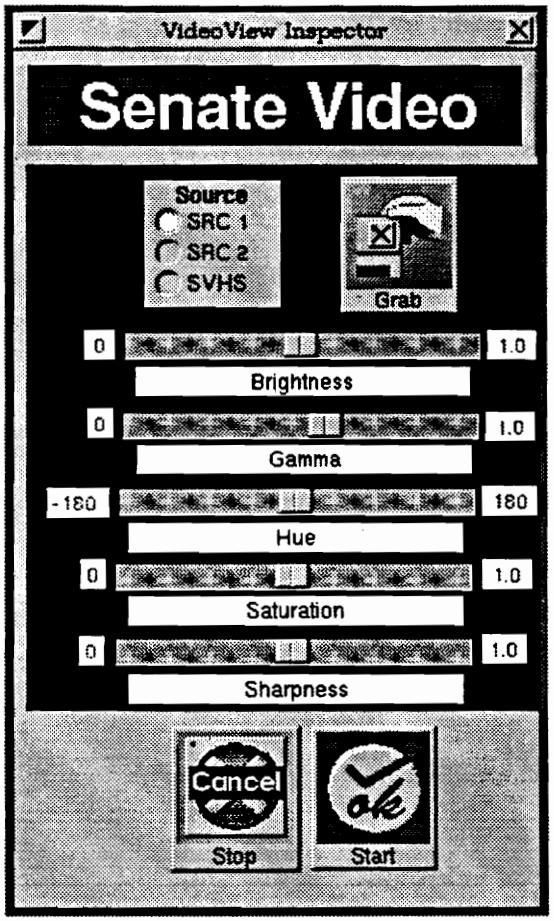


Figure 4.28 Full Motion Video Support Panel

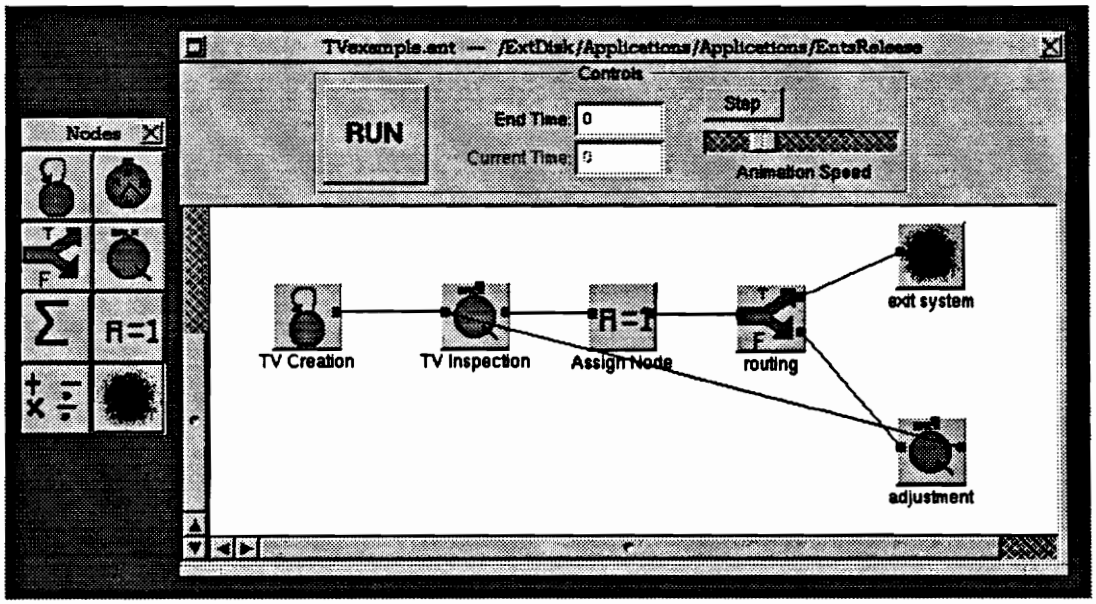


Figure 4.29 NeXT Ents Application Screen

4.9 Edit Menu

The Edit command attaches the Edit menu, which contains commands affecting the current selection in any editable documents or selectable text. Figure 4.30 shows the Edit menu.

The Edit menu contains the commands that alter the selection in the current key window. Each command is dimmed when it cannot operate on a current selection. the following is a brief description of the various Edit commands.

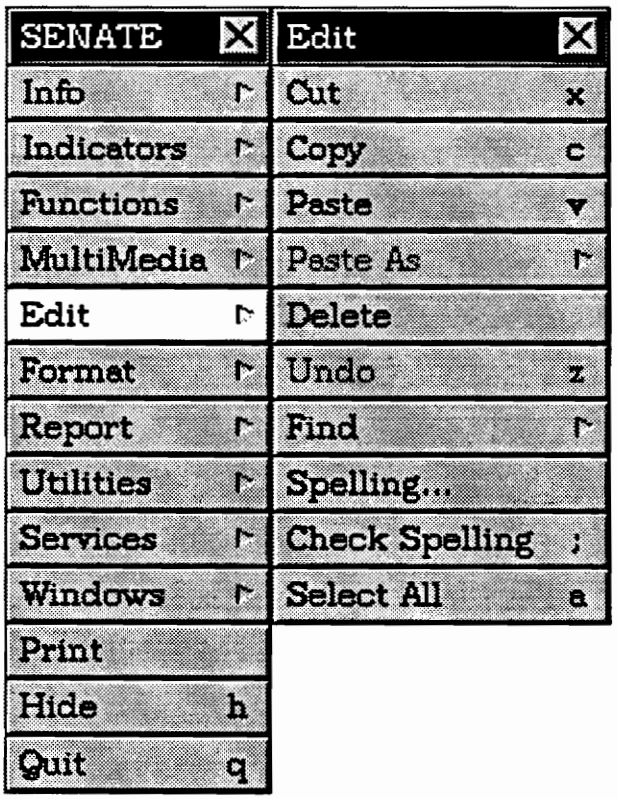


Figure 4.30 Edit Menu

Command	**Action**
Cut	Deletes the current selection and copies it to the pasteboard.
Copy	Copies the current selection to the pasteboard without deleting it.
Paste	Replaces the current selection with the contents of the pasteboard.
Paste As	Attaches a submenu that permits the user to paste the current contents of the

pasteboard into the document in a specified data type.

Delete — Deletes the current selection without copying it to the pasteboard (thus leaving the contents of the pasteboard intact). The Delete key has the same effect.

Find — Attaches the Find menu, which contains commands related to the Find panel.

Command	Action
Find Panel...	Brings up the Find panel, makes it the key window, and selects everything in the text field labeled Find so that the user can easily enter new text. If the panel is already on-screen, the command brings it to the front, makes it the key window, and selects the Find field.
Find Next	Searches forwards for the next occurrence of the string in the panel's Find field.
Find Previous	Searches backwards for the previous occurrence of the string in the panel's Find field.
Enter Selection	Enters the current selection into the panel's Find field so that Find Next and Find Previous can search for it.
Jump to Selection	Scrolls to display the beginning of the current selection.

Find Next and Find Previous begin searching at the current selection. If the search is successful, the text found is selected and becomes the starting point for the subsequent search. Neither command requires the Find panel to be on-screen. However, if the panel's Find field is empty, Find Next and Find Previous both bring up the Find panel, make it the key window, and select its Find field. This is exactly what the Find Panel command does. These other commands do it as a convenience to the user, who has indicated an intention to do a search.

Spelling...	Brings up the Spelling panel.
Check Spelling	Finds the next misspelled word without bringing up the Spelling panel.
Select All	Makes the entire contents of the file the current selection.

4.10 Format Menu

The Format command attaches the Format menu, which contains commands affecting the layout of documents, including the font and paragraph format of text and the arrangement of graphic images. Figure 4.31 shows the Format menu.

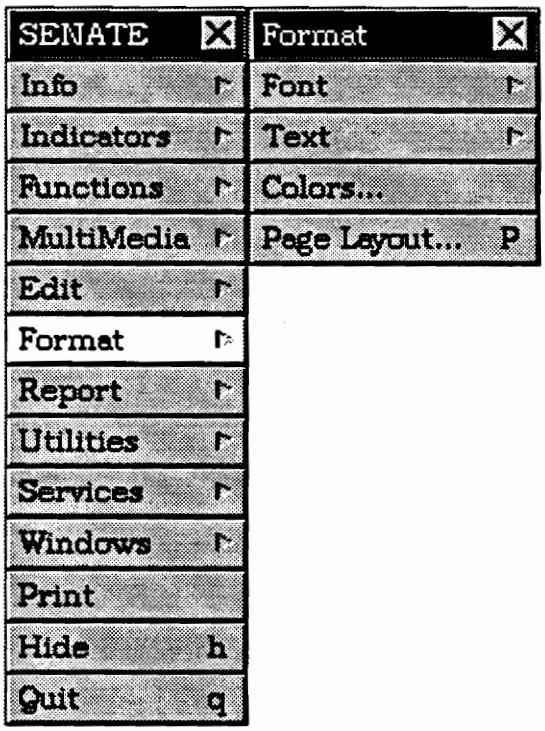


Figure 4.31 Format Menu

Command	Action
Font	Brings up the Font menu, which has commands to alter the font of the current selection. It contains the following commands:

Command	Action
Font Panel...	Brings up the Font panel.
Bold	Makes the current selection bold, if it's not bold already, and makes it unbold if it is. The name of the command alternates between Bold and Unbold depending on the selection.
Italic	Makes the current selection italic or oblique, if it isn't already, and makes it unitalic if it is. The name of the command alternates between Italic and Unitalic depending on the selection.
Underline	Underlines the current selection, if it isn't already underlined, and removes the underlining if it is. When the current selection is already underlined, the command name must change to Ununderline.
Larger	Makes the current selection one point larger.
Smaller	Makes the current selection one point smaller.
Heavier	Uses a heavier typeface to display the current selection.
Lighter	Uses a lighter typeface to display the current selection.
Superscript	Moves the currently selected text up an appropriate amount for a superscript. Choosing the command again moves the text that much higher.
Subscript	Moves the currently selected text down an appropriate amount for a subscript. Choosing the command again moves the text that much lower.
Unscript	Returns the selected superscripted or subscripted text to the normal baseline of the text.

Text — Attaches the Text menu, which lets the user choose the format of the selected blocks of text. All the Text menu commands are supported by the

Application Kit's Text object. It contains the following commands:

Command	Action
Align Left	Aligns the text at the left margin, leaving a ragged right margin.
Center	Centers the text between the left and right margins.
Align Right	Aligns the text at the right margin, leaving a ragged left margin.
Justify	Aligns the text at both the left and right margins.
Show Ruler	Displays a ruler in the text area, if the ruler isn't currently visible. Otherwise, this command hides the ruler. The name must alternates between Show Ruler and Hide Ruler, depending on the state of the text area. The ruler is a scale containing controls that affect the format of a paragraph (such as margins and tabs).
Copy Ruler	Copies the ruler settings in the first paragraph of the selected text.
Paste Ruler	Alters the paragraphs containing the text selection to have the settings most recently copied with the Copy Ruler command.

Colors The Colors command brings up the Colors panel. This panel is provided by the Application Kit. It lets the user preview and specify colors in any of the following modes: color wheel, grayscale, red-green-blue (RGB), cyan-magenta-yellow-black (CMYK), hue-saturation-brightness (HSB), custom palette (which loads an image from which the user can choose colors), and custom color lists.

Page Layout... Brings up the Page Layout panel, which lets users determine how documents

are to be printed and displayed on the screen.

4.11 Report Menu

The SENATE Report Menu is shown in the following figure:

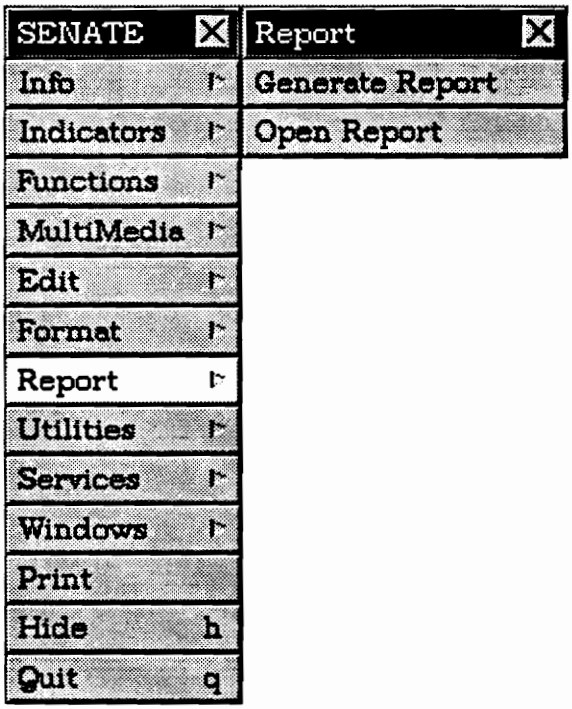


Figure 4.32 Report Menu

The Report Menu has the following commands:

Command	Action
Generate Report	This command generates the simulation study credibility assessment report for a particular user and project. This command should be used only after saving your session using the Save Indicators command in the Indicators Menu. It is stored as an ASCII file in: ~/kbess_rpt/<user_name>_<project_name>.rpt file.

Open Report	This command opens the report generated by the Generate Report command. The /NextApps/Edit.app application displays the file : ~/kbess_rpt/<user_name>_<project_name>.rpt . You can print this file using the Edit.app Print command. To return to SENATE, quit Edit.app

4.12 Utilities Menu

The Utilities Menu contains the following options [Figure 4.33]:

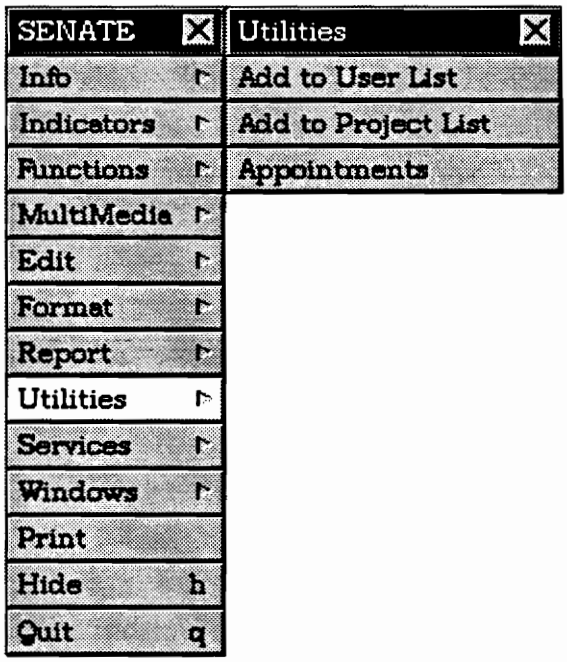


Figure 4.33 Utilities Menu

The Add to User List and Add to Project List commands let an operator in the Administrator Mode add new user and project names to the User Browser and Project Browser respectively.

The third option, Appointments executes a NeXT application Date. Date is an electronic Datebook application by Brian Yamamoto at NeXT Computer Inc. The

following is a brief description of the software by its author.

There are 5 views a datebook can show: the year, month, day, appointment, and events view. The initial view a datebook shows is the month view. In the month view, double clicking a day will open up a day view. Once in the day view, double clicking on an hour will open up the appointment view. If you press the Return key while in the appointment view, you will return to the day view. Pressing the return key again will return you to the month view. Pressing the return key again will show you the year view.

You can get the events view by choosing the Show Events command in the Window menu. This view shows all the appointment that are in the datebook. Pressing the Return key will return you back to the month view.

Once the software is enabled, you can create appointments and reminders in your datebook. The easiest way to create a new appointment is to choose the New Event command in the Window menu, which will bring up the appointment view. If you fill in the What: and When: fields, then press the Return key, you've created your first appointment.

When you edit any field of the appointment view, the datebook window shows that it is being edited by changing the icon in the window's close button. If you want to revert all changes to the original state, choose the Revert To Saved command in the Window menu.

You can create a reminder for any appointment. The remind may be a window reminder, a mail reminder, or both. First, open an appointment in an appointment view. With the Reminder Type pop-up list, choose the type of reminder you want. Fill the the Remind At: field with the amount of time before the appointment that you want your reminder to occur. For example, if you wanted a reminder to be sent a day before the appointment, you would set the Remind At: field to 1, and click the Days item as the unit of time.

If you've chosen a Mail reminder, type in the login names of the people that you want to receive the reminder. Each name should be separated by a comma. You don't need to enter your own name, you will automatically receive all Mail reminders.

Some appointments in your datebook will be repeating appointments, like weekly meeting or monthly bill or holidays like Christmas and Thanksgiving. To make an appointment repeat, choose the repeating frequency of the appointment in the Frequence pop-up list. You can also optionally limit a repeating appointment with a Start Day and an End Day.

Here are the meanings of the Frequency pop-up list: The Daily, Weekly, Bi-Weekly, Monthly, Bi-Monthly, and Yearly items make the appointment repeat at the interval the title suggests. The Nth Weekday item is to get appointments like "the 2nd Tuesday of the month" or "the 3rd Wednesday of the month". The Last Weekday item is to get appointments like "the last Saturday of the month". The Last Day item will cause the event to occur on the last day of the month.

You can use the Report command in the Window menu to create a detailed report. The Report command brings up a Report panel. After setting the Start Day and End Day fields, click the Save button to save the report into a file. The Print button will save the report into a temporary file, and then tell the Edit application to print the file.

In the year, month, and day views, you can use the Previous and Next button to see the previous or next year, month or day. You can also use the Find panel to go to a specific date without using the Previous or Next buttons.

The SENATE datebook is placed in the file ~/kbess_multimedia/Active.datebk file. This application is very helpful for keeping track of the evaluation sessions the users have scheduled and also for sending reminders. Date application is in /LocalApps.

4.13 Services Menu

This menu contains commands that invoke services provided by other applications on the machine such as Webster's Dictionary, Grab, Edit, Terminal etc.

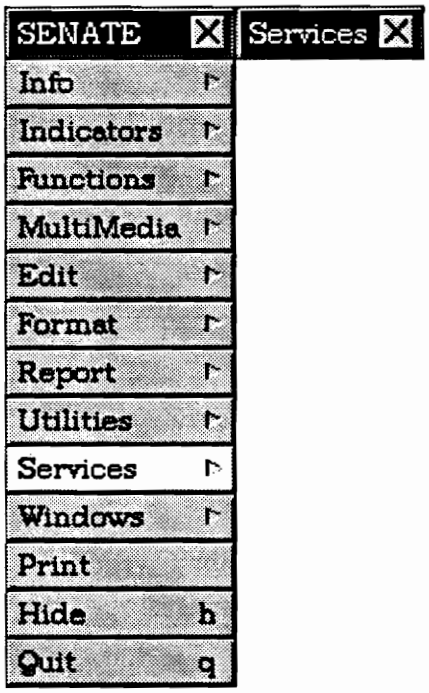


Figure 4.34 Services Menu

4.14 Windows Menu

The Windows menu contains commands affecting the windows that belong to the application.

Command	Action
Arrange in Front	Stacks and offsets all the application's document windows
Miniaturize Window	Miniaturizes the key window (if it has a miniaturize button). The affected window need not be a document window.
Close Window	Closes the key window (if it has a close button). If the window is the last one (or only one) open displaying a document, it also closes the document, just as the Close command would.

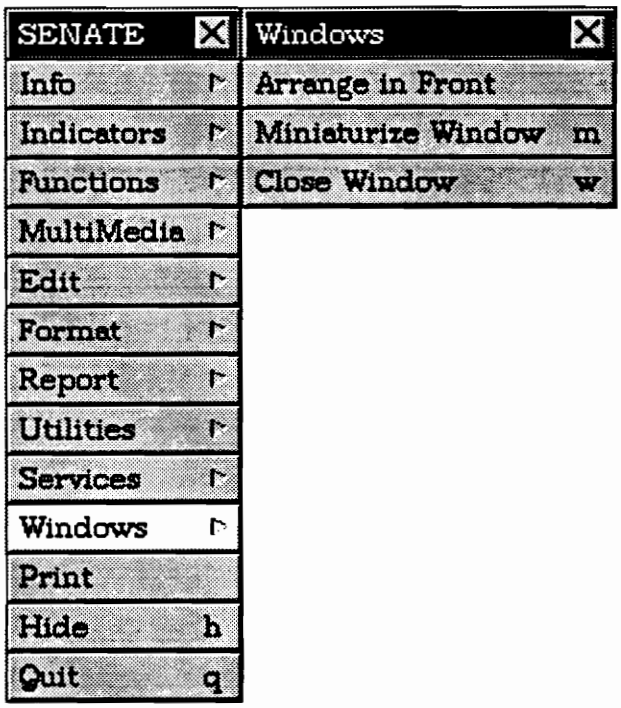


Figure 4.35 Windows Menu

4.15 Print Command

The Print command brings up a Print panel [Figure 4.36]. This panel is an attention panel that's provided by the application kit. This panel comes up every time the user wants to print a document or other data. After specifying the information needed for printing, the user can do any of the following: send the output to a printer; save the output to a PostScript file, instead of printing it; send the output to a fax modem, instead of a printer; preview on-screen what will be printed; cancel any of the above selections, even after they've started.

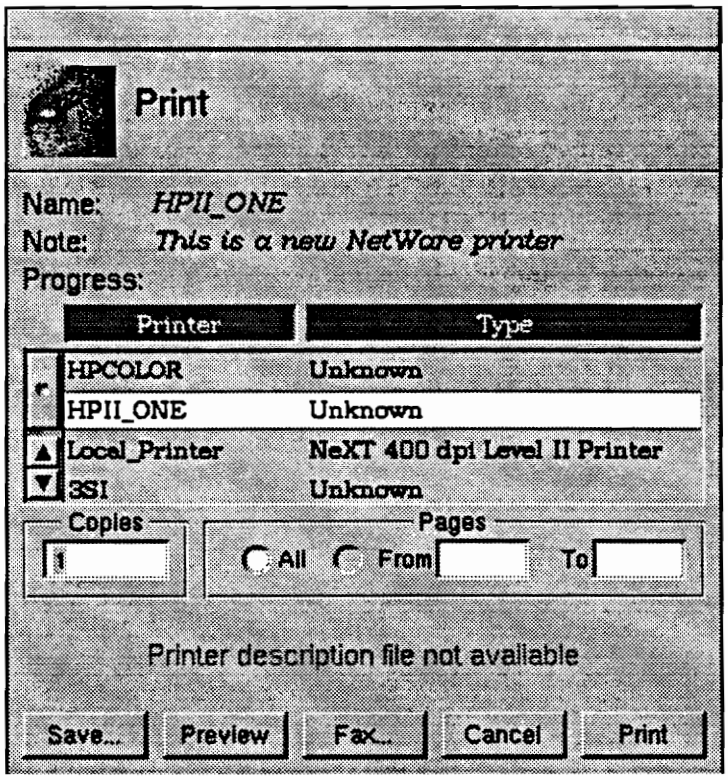


Figure 4.36 Print Panel

4. 16 Hide Command

The Hide menu command lets the user clear the screen of all the windows belonging to an application. This opens up the workspace so that it's easier to work in another application.

When the application is hidden, only its application icon remains on-screen. When the user double-clicks the icon, the hidden windows reappear on-screen. Users can resume working in the application, picking up again at exactly the point where they left off. Double-clicking an application icon has one other effect: It activates the application, and so may cause the menus and panels of another application to disappear, while those of the newly activated application reappear.

4.17 Quit Command

Quit terminates the application. Quitting SENATE without saving a session might cause the user to lose work, therefore the application brings up a Quit panel, which requires the user to confirm a Quit command [Figure 4.37]. The user can click on the Cancel button to return back to the session.

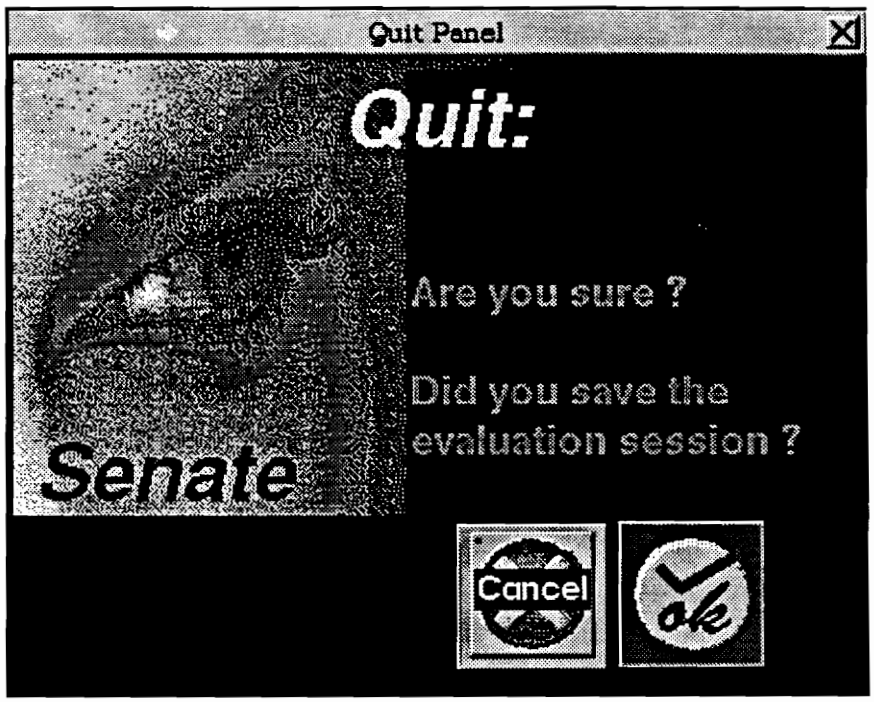


Figure 4.37 Quit Panel

Chapter 5
Recommendations for Future Research

SENATE has been prototyped by using the evolutionary software development approach. Because of this, a variety of characteristics of SENATE can stand improvement. There are some important enhancements that have been identified and deemed important but have not been incorporated into to the current version of the system. They are:

- The integration of a relational database management system (RDBMS) for retaining and processing the expert evaluation knowledge would greatly increase the efficiency of the system. Basic support for the SYBASE RDBMS has been provided in the current system.
- It is important to provide preventive features in a system such as SENATE. It is important to provide a universal undo command that will help users recover from mistakes.
- At present SENATE has a context sensitive help system with a online user manual. But for the system to be more effective it needs to have some brief tutorials and examples that would help users use the system more effectively.
- We have provided a very basic support for multimedia components in the system. Integration with highly sophisticated multimedia systems would greatly enhance the effectiveness of the system.
- Provide capabilities to view the indicators not just in a browser view but also as graphs or trees, where each indicator may be an icon (node) in the tree or use similar visualization techniques.
- Integration with a knowledge base would offer great potential for improving the confidence in the assessment of the credibility of a simulation study. For creating this knowledge base it will be required to implement a basic expert system shell that would let the system administrators build the rules for a particular project. These rules will identify and define the dependencies

between the indicators in the hierarchy. The engine should be capable of prescribing remedies for inconsistencies and errors It would rely on its knowledge to create specifications or recommendations for correcting a diagnosed problem. The inference engine based on these rules would make appropriate inferences, in order to increase the efficiency and productivity of the expert peers, and for providing a useful solution to the sponsor. Hence, knowledge representation and inference techniques of a knowledge based system would greatly enhance the SENATE system.

References

Balci, O. (1989), "How to Assess the Acceptability and Credibility of Simulation Results." *Proceedings of Winter Simulation Conference*, IEEE, Picataway, NJ, pp. 62-71.

Balci, O. (1987), *Guidelines for Successful Simulation Studies.* Technical Report TR-85-2, Department of Computer Science, VPI&SU, Blacksburg, VA (March).

Balci, O. (1986), "Requirements of Model Development Environments." *Computers & Operations Research* 13, 1 (Jan.-Feb.), 53-67.

Balci, O. and R .E. Nance (1985), "Formulated Problem Verification as an Explicit Requirement of Model Credibility." *Simulation* 45, 2 (Aug.), 76-86.

Balci, O. and R.E. Nance (1992), "The SImulation Model Development Environment: An Overview." *Proceedings of the 1992 Winter Simulation Conference*, IEEE, Piscataway, NJ, pp. 726-736.

Elmaghraby, S.E. (1968), "The Role of Modeling in IE Design." *Industrial Engineering* 19, 6 (June), 292-305.

NeXT Computer, Inc. (1992a), *NeXTSTEP User Interface Guidelines.* Addison Wesley Publ. Co., Reading, MA.

NeXT Computer, Inc. (1992b), *NeXTSTEP General Reference Vol.1.* Addison Wesley Publ. Co., Reading, MA.

NeXT Computer, Inc. (1992c), *NeXTSTEP General Reference Vol.2.* Addison Wesley Publ. Co., Reading, MA.

NeXT Computer, Inc. (1992d), *NeXTSTEP Development Tools and Techniques.* Addison Wesley Publ. Co., Reading, MA.

NeXT Computer, Inc. (1991), *NeXTSTEP Advantage.* NeXT Publications, Redwood City, CA.

Nunnally, J.C. (1978), *Psychometric Theory.* McGraw Hill, New York, NY.

Pinson, L.J., Wiener, R.S., (1991), *Objective C: Programming Techniques.* Addison Wesley Publ. Co., Reading, MA.

Webster, B.F., (1989), *The NeXT Book.* Addison Wesley Publ. Co., Reading, MA.