# A Prototype Assistance Manager For The Simulation Model Development Environment

*Valerie L. Frankel*
*Osman Balci*

TR 87-21

Technical Report 87-21†

# A PROTOTYPE ASSISTANCE MANAGER FOR THE SIMULATION MODEL DEVELOPMENT ENVIRONMENT

Valerie L. Frankel
Osman Balci

Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, Virginia 24061

1 August 1987

† Also cross referenced as SRC-87-009, Systems Research Center.

## ABSTRACT

The Assistance Manager, one of the tools of the Simulation Model Development Environment (SMDE), is required to provide effective and efficient transfer of assistance information to an SMDE user. This paper describes a prototype of the SMDE Assistance Manager. Objectives are set forth and a design is established and implemented on a SUN 3/160C workstation. The prototype is evaluated with respect to the design objectives and is shown to provide a highly flexible interface between the user and the database of assistance information. Results indicate that the prototype Assistance Manager incorporates the characteristics considered desirable in on-line assistance systems and serves as a basis for future enhancement and development.



**CR Categories and Subject Descriptors:** I.6.m [**Simulation and Modeling**]: Miscellaneous

**General Terms:** Design, Human Factors

**Additional Key Words and Phrases:** Model development environments, on-line assistance, simulation support systems.

# 1. INTRODUCTION

A prototype simulation model development environment (SMDE) has been developed to provide an integrated and comprehensive collection of computer-based tools to: (1) offer cost-effective, integrated, and automated support of model development throughout its entire life cycle, (2) improve the model quality by effectively assisting in the quality assurance of the model, (3) significantly increase the efficiency and productivity of the project team, and (4) substantially decrease the model development time [Balci 1986; Balci and Nance 1987a,b].

Large systems, such as the SMDE, are usually difficult to comprehend and use. Significant effort is needed to develop an assistance system that would alleviate many of the usability problems, as well as enhance the ease of including assistance in new SMDE components.

Because the SMDE is a general purpose environment that can support a variety of modeling and simulation applications, it is not unreasonable to expect that it may be used by people with widely different backgrounds and expertise. For a modeler to work productively and confidently within the SMDE, great care must be given to design an interface that is integrated with an on-line assistance system. This system, to be effective, is required to be constantly available, easy to use, and sensitive to the user's current state. Furthermore, these requirements must be met by a help system that does not dominate the user's interactions within the environment. It must be flexible enough to accommodate varying levels of user expertise, and should be accessible from any tool in the SMDE. Because the SMDE is an open-ended environment, the help system must be general and flexible enough to be easily incorporated into existing tools, as well as included with new tools which may be added to the environment later.

Existing help systems are ill-equipped to handle these requirements. Current help

systems available for general use are add-on systems designed to operate independently of the system about which help is provided. They lack the flexibility and context sensitivity necessary for use within the SMDE.

The purpose of this paper is to describe the prototype Assistance Manager tool of the SMDE. Section 2 introduces on-line assistance and identifies desirable characteristics of effective assistance systems. The design objectives, hardware/software environment, and components of the Assistance Manager are described in Section 3. Section 4 contains the evaluation of the prototype and conclusions are stated in Section 5.

## 2. ON-LINE ASSISTANCE

An on-line assistance system is a software system which aids a user in using some software facility while the user is directly connected with the computer. This aid may be offered in the form of command assistance, error recovery, on-line documentation, or computer-aided instruction. The assistance may be separate from the system on which help is offered, or it may be integrated closely with the system architecture.

### 2.1 Background

It is now standard practice to include some form of on-line assistance with an interactive system. From the user's point of view, the major differences in these help systems are in the kind of help offered, and the mechanisms used to access the information.

Many researchers who characterize their work as "on-line help" consider their design in terms of users recovering from errors. These systems are concerned with context sensitivity; i.e., how the system can find out enough about what the user was doing to explain the problem and offer advice. People who characterize their work as "on-line documentation" typically think of making reference material available to the user on request [Walker 1986].

When the amount and depth of information needed to start using a system is more than can be addressed by a help system, on-line tutors can be used. Tutors have the advantage of offering a new user a safe, restricted environment where he can learn and subsequently test his knowledge. A familiar example of an on-line tutorial is the "learn" command available on many UNIX† systems [Kernighan and Pike 1984].

With the advent of personal computers, computing became accessible to a diverse population. Interactive systems became more widespread and it was no longer possible to rely on intuition and ad hoc methods in designing help systems. Help began to be integrated: developed around — and not in spite of — the software system itself. Examples of integrated help systems are systems that base their user interface on menus with well-described choices [Bergman and Keene-Moore 1985].

Even the best help system would be useless without a well designed interface. In addition, requirements for a help system may in large part determine the user interface. For this reason, basic principles of interface design must be explored.

User interface design is more accurately termed art than science. Often, there is no "right" answer, or no scientific basis to justify one choice over another. However, empirical studies can provide guidance for the designer.

The designers of the Xerox STAR system [Smith et al. 1982], who pioneered user interface mechanisms like icons and windows, advocate consistency and simplicity throughout. Mechanisms should be used in the same way wherever they occur. Different modes of interaction, in which input may mean one thing in one mode and something entirely different in another, are to be avoided.

Some researchers [Yestingsmeier 1984] advocate early user involvement in interface design, maintaining that users can best describe their needs and expectations from a

† UNIX is a trademark of Bell Laboratories.

system. Others [Blake 1986] assert that blindly implementing user preferences can be disastrous.

Other user interface issues which have relevance in the design of on-line assistance are: (1) the use of menus [Hodgson and Ruth 1985], (2) designing systems which minimize user errors [Norman 1983], (3) choice of input mechanisms [Card et al. 1978], (4) window layout and design [Gait 1985], and (5) "soft-machine" interfaces [Nakatani 1983].

Research has been done on issues in help system design as well [Connolly et al. 1985; Houghton 1984]. More often than not, researchers have started from scratch, only to "rediscover" the basic paradigms. Borenstein [1985] warns that the greatest problem affecting builders of help systems is lack of knowledge of what has already been done. Anyone who wishes to implement a help system must study past and present systems to avoid making common mistakes.

In this research, we studied the following help systems and Frankel [1987] provides their survey: ACRONYM [Borenstein 1985]; AT&T Unix System Help Facility [Butler and Kennedy 1985]; BROWSE [Bramwell 1983]; DOMAIN/DELPHI [Orwick et al. 1986]; HELP: A Question Answering System [Roberts 1970]; ICNHELP [Stoddard et al. 1985]; SARA [Estrin et al. 1986]; SIGMA [Rothenberg 1979]; TNT [Nakatani et al. 1986]; UC: The Unix Consultant [Chin 1986]; and ZOG [Robertson et al. 1981].

## 2.2 Characteristics of Effective On-line Assistance

Previous research has identified characteristics, described below, considered desirable for effective assistance systems [Relles et al. 1981; Smith et al. 1982; Walker 1986; Sondheimer et al. 1982].

*Completeness* of textual material is required if the user is to have faith in the assistance system. Incomplete help can be as bad as no help at all, leaving the user frustrated and confused as to what action to take.

*Consistency* is important both in accessing help and in the conventions used to present help text, prompts, and diagnostics. Assistance should be requested in a similar manner in all of the interactive programs that make up a larger system.

*Context sensitivity* refers to the system's ability to provide help relevant to the current situation. This implies that the system is keeping track of the user's current state of interaction. The main importance of context sensitive help is that the user does not have to deal with information that is irrelevant or inapplicable.

*Expandability* enables future growth for an open-ended environment. Related to expandability is maintainability. It is a must that the system be easily updated to keep the help texts current and complete.

*Flexibility* implies that different levels of user expertise are accounted for in the presentation, composition, and access of help information. The system should be terse enough for more experienced users, yet simple enough for beginners to quickly become confident in their use of the system.

*Understandability* is achieved by use of clear, concise language, use of technical terms only when necessary, and the judicious use of such display features as highlighting and text chunking.

*Unobtrusiveness* refers to the ability to request assistance without interrupting the task at hand. Too often, the user must explicitly save his state and then restore it after help is obtained. Unobtrusiveness also deals with the way help is accessed. Help should be immediately available, but not noticed until it is needed.

These characteristics are included as part of the requirements for the SMDE Assistance Manager. Many are "common sense" in nature, and it is intuitively obvious why these characteristics would be considered desirable. In spite of this, few if any help systems demonstrate all of these traits in more than a cursory way. The reasons for the lack

of commitment vary from a failure to recognize the importance of on-line assistance, to insufficient resources for their design and implementation. By recognizing their importance early in the design phase, these features can be used to guide design decisions.

## 3. PROTOTYPE ASSISTANCE MANAGER

This section describes the design objectives, hardware/software environment, and components of the prototype SMDE Assistance Manager.

### 3.1 Design Objectives

The overall goal of the Assistance Manager is to provide effective and efficient transfer of assistance information to an SMDE user. "Effective" means accurate information is provided which is relevant to the user's needs. "Efficient" implies that if the user is involved in interaction with the SMDE, it is not necessary to switch tasks or modes in the process of seeking help. The objectives identified below are intended to meet this overall goal.

(1) Provide general information for beginning system users. Such information would serve to acquaint new users with the environment, and establish a context for subsequent learning.

(2) Provide detailed and specific help on the use of an SMDE tool.

(3) Provide definitions and example usages of technical terms encountered in documentation and communication within the environment.

(4) Provide tutorial assistance for SMDE users. The tutorial should give the user a protected arena for limited experimentation with a tool's features.

(5) Provide help that is constantly available and immediately accessible. There should be methods to temporarily suspend interaction with the Assistance Manager, or save the current display for future reference. The user should not be required to step through an artificial protocol or syntax to access immediate assistance.

(6) Provide help that is unobtrusive; i.e., messages or prompts that are only visible when required or asked for.

(7) Provide a help system that is flexible enough to accommodate experienced users as well as novice or casual users.

(8) Provide context-sensitive help wherever possible. The system should use all available information on the user's state and avoid placing the burden on the user.

(9) Provide appropriate methods of access to the help information. Initiative, mechanisms, and complexity of access should vary according to task and type of user.

(10) Provide a straightforward and systematic method for tool developers (application programmers) to build help into tools which may be added to the environment.

(11) Provide help that is available in a consistent manner from any tool within the environment.

(12) Administer the assistance database by serving as an interface between the user or programmer and the database contents.

(13) Provide easy methods for update and expansion of the Assistance Manager database. This is critical in order to accommodate the tailoring and updates that are inevitable in a large software environment. Updates should be enforced in a manner which helps enforce database integrity and consistency.

### 3.2 Hardware/Software Environment

The architecture of choice for the prototyping of the SMDE, and in particular the Assistance Manager, was a SUN 3/160C† color workstation running a Berkeley 4.2 based UNIX operating system. The SUN is a 2-MIPS machine consisting of a 16.67-MHz MC68020 microprocessor with a 16.67-MHz MC68881 floating-point co-processor, a 380-MB Fujitsu Eagle disk subsystem, a 1/4-inch cartridge tape subsystem, 4-MB of main memory, a 19-inch color monitor with a resolution of $1152 \times 900$ pixels, a pointing device called a mouse, and a connection to the Ethernet network which enables high speed file transfer to and from other University computing systems.

The most important ingredient of the user interface is the 19-inch bit-mapped display screen. Because every screen pixel can be turned on and off, the SUN has an excellent ability to present visual images. Much of the work done in this research dealt

† SUN 3/160C is a trademark of Sun Microsystems, Inc.

with this feature of the system; i.e., the user interface capabilities.

The user communicates with the SUN by keyboard input. In addition, the mouse can be used to point to or select locations on the screen. The system provides continuous feedback as to where the mouse is pointing by displaying a cursor on the screen. Virtually any material that is displayed on the screen can be pointed at and treated as input.

The user views the environment through a display consisting of rectangular "windows". Windows are self-contained work areas and are analogous to sheets of paper on a desk top. They can be overlapped on the screen, effectively increasing the user's working space. Windows may be resized, "closed" or collapsed to a small figure commonly called an icon, and subsequently "reopened". Within a window, text may be scrolled for viewing.

Each window corresponds to a different task or aspect of the user's environment. A user can switch back and forth between tasks, which is of great value when the tasks are related in some way. Thus, a user may be editing a file on one window, running a program in another, checking a file listing in yet another, etc. The SUN's window management system (SunView) takes care of updating the display image, tracking user input, and similar concerns.

One technique heavily used throughout the system is the use of menus. Menus facilitate context switching between tasks because the user does not have to remember a command. Rather, he may use the mouse to select one of a finite set of options on a menu. Often, the menus are "pop-up", appearing only as a result of depressing a button on the mouse. This allows for conservation of screen space and a high degree of unobtrusiveness. Menu options are frequently selected by a software "button", a button-like image over which the user can position the mouse cursor and "select" by depressing a button on the mouse.

The Assistance Manager consists of approximately 4500 lines of documented code. The vast majority of this code is written in the C Programming Language. Some parts dealing with file and directory manipulation are written using the UNIX *shell, sed,* and *awk* facilities [Kernighan and Pike 1984].

Window features are programmed using the *SunView* application package, which consists of high-level routines to create visually communicative interfaces and manage the window system [Sun Microsystems 1986a].

Databases are created using the INGRES relational database management system [Sun Microsystems 1986b]. EQUEL (Embedded QUEry Language)/C is used in C programs to access INGRES. Storing comments and adding data to the help database are achieved by using the INGRES/FORMS Application Package.

Without the hardware and software configuration just described, it would have been impossible to design an Assistance Manager which could hope to embody the desirable characteristics described in Section 2.2.

### 3.3 Components of the Assistance Manager

The Assistance Manager may be run as an independent tool by invoking it through the UNIX shell. Entering the command "am_menu" in the console window results in the display of the Assistance Manager's top-level menu as shown in Figure 1. By enabling access to the Assistance Manager at the UNIX command level, a user may learn about the SMDE without actually entering it.

The Assistance Manager may also be invoked by selecting from a pop-up menu available on any other SMDE tool. For example, by entering "moddef" on the terminal window, Model Generator can be invoked and by depressing the third button of the mouse on the help menu item, the help pop-up menu can be displayed from which Assistance Manager can be activated as depicted in Figure 1. In this case, the Assistance

**Figure 1.** Activation of the Assistance Manager.

Manager is spawned as a separate process.

Once the Assistance Manager has been invoked, a user may select any of its components by depressing the first mouse button on the icon of the component desired. Because each component is spawned as a separate process, it is possible to run many of them simultaneously. When not in use, the Assistance Manager or its component may be collapsed into an icon to conserve screen space. The icon at the top-right corner of Figure 1 is another Assistance Manager process which can be activated by depressing the first mouse button on it.

### *3.3.1 Introduction to the SMDE*

This component provides general information for beginning users of the environment. It discusses SMDE tool features and uses. In this sense, it may be considered as the on-line documentation component of the Assistance Manager, although it makes no attempt to serve as a reference manual. The Introduction gives references for the curious user who wishes to find out more about the concepts and methodologies that underlie the SMDE.

The interface to the Introduction consists of two viewing windows and one control panel as depicted in the "Introduction to MDE" window of Figure 2. One viewing window is provided for text, and another for the table of contents of the text being viewed. Both viewing windows may be scrolled up or down by using the mouse, and either window may be further split into multiple views of the same text. (See the pop-up menu of Figure 3 for options of manipulating a scroll window.)

The control panel allows the user to select a topic from the table of contents for display, or to search for a particular pattern. For example, in the partially-covered "Introduction to MDE" window, the search for the "Conical Methodology" pattern is illustrated. A "Show Documents" selectable menu item is provided to return the user to

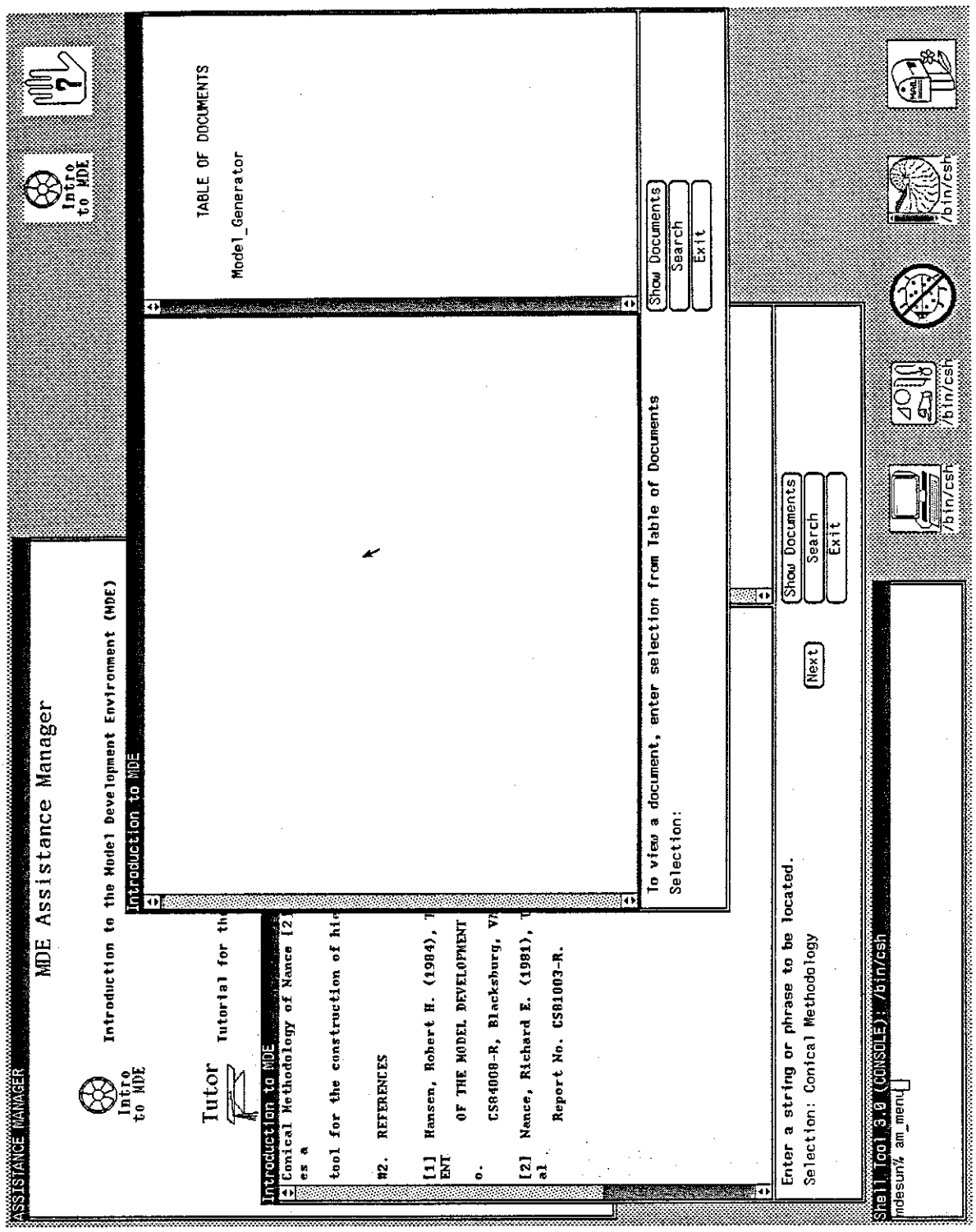


**Figure 2.** The *Introduction to the SMDE* component.

the top level table of contents, which lists all chapters available for viewing. An exit menu item is provided to leave the Introduction. (The user can also exit, if desired, by using the standard pop-up menu on the tool border stripe.)

The Assistance Manager provides the mechanism by which documents are updated or added to the documents database. The document writer develops and formats the text for display using a text editor and/or formatter. Once the text is in a displayable format, the writer adds the document to the documents database by accessing the Assistance Manager and selecting "Programmer Assistance". This feature is explored in Section 3.3.6.

### *3.3.2 Tutorial*

The Tutorial component, like the Introduction to the SMDE component, is made up of an interface and a database of documents. The documents, however, will differ considerably from the type used in the Introduction. Whereas the Introduction tells what the SMDE is all about, the Tutorial deals with how to use the SMDE tools.

The interface to the Tutorial consists of a viewing window and a control panel as shown in Figure 3. The viewing window displays either a list of available tutorials or the text of a tutorial which the user has selected. The text in the viewing window may be scrolled up or down using the mouse and may be further split to allow multiple views of the text. Other options for manipulating a scroll window can be selected from the pop-up menu of Figure 3.

The control panel consists of a status prompt and menu items which allow the user to list all available tutorials, search for a string, phrase, or topic in the text, or exit from the Tutorial. The Tutorial runs as a separate process and may be invoked either from the top level Assistance Manager menu or from an SMDE tool. When not in use, it may be exited or closed to an icon to conserve screen space.

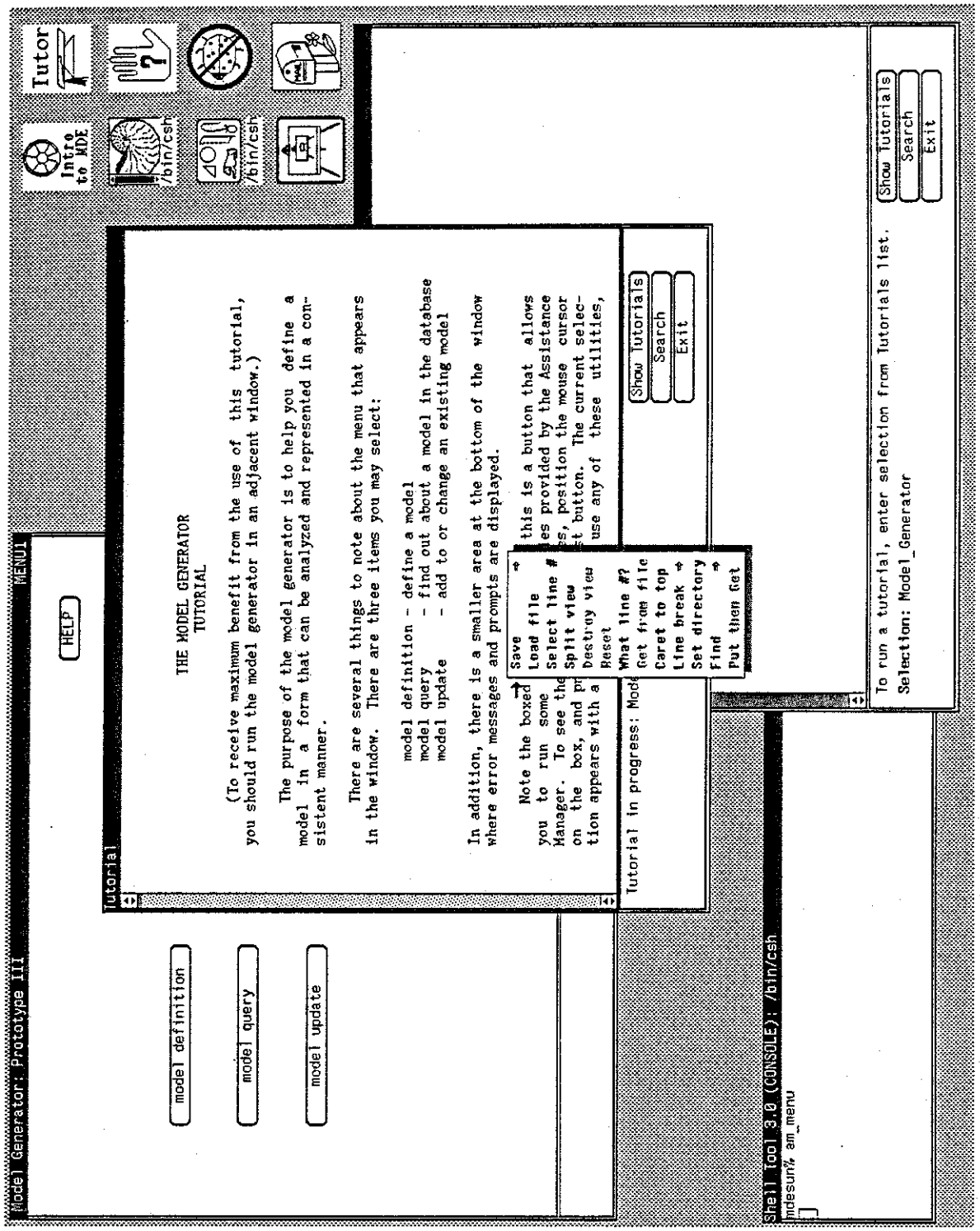


**Figure 3.** The *Tutorial* component.

Ideally, the Tutorial is run in a window alongside the tool of interest, giving the user step-by-step instructions on how to proceed. This approach gives the interactive behavior necessary for effective tutoring, yet removes the burden of interaction from the Tutorial itself.

The Tutorial for an SMDE tool should be created by the tool developer or someone else who is knowledgeable in the use of the tool. The text of the tutorial is prepared with a text editor and/or formatter. Once the tutorial is in a suitable format, it is added to the tutorial database in a manner similar to that used for the Introduction documents.

### *3.3.3 Glossary*

The Assistance Manager provides a glossary feature which can be used to find the meaning of a technical term or a term which has a definition specific to an application. The glossary runs in its own window and interfaces directly with the glossary database. Like the other functions which can be chosen from the Assistance Manager menu, the glossary may also be invoked from a tool within the SMDE and runs independently of the calling process. When not in use, it may be exited or collapsed to an icon to conserve screen space.

As shown in menu 1 of Figure 4, a user calling the glossary is confronted by a window containing a status area and selectable menu items of browse, lookup, and exit. The lookup option provides direct access to a term in the glossary. If the user selects lookup, the menu is replaced by menu 2A which first prompts for the term in question, and then displays its definition and an example of its usage or meaning. If the term is not contained in the glossary, a message to that effect is displayed. The status area instructs the user on window use. At this level, the user may return to the main menu, exit the glossary, or collapse the glossary into an icon.

The user who wishes to browse through the glossary may select the browse option

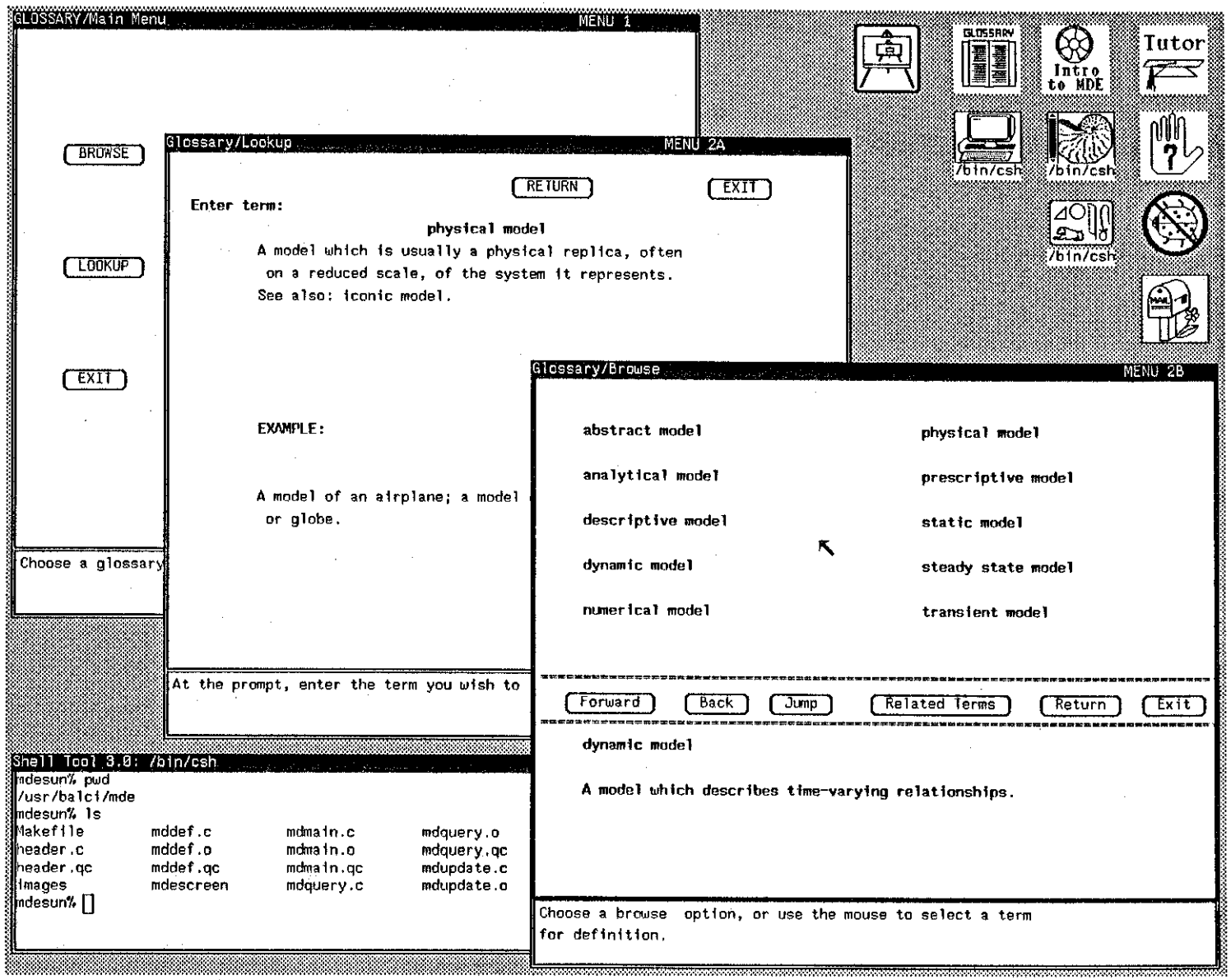


Figure 4. The *Glossary* component.

from the glossary main menu and obtain menu 2B of Figure 4. Menu 2B is considerably more complex, in keeping with the more advanced functionality provided, and contains a display area, a control panel, a viewing area, and a status area.

By default, the first ten terms in the glossary appear immediately in the display area. The user selects "Forward" or "Back" to view additional terms. To display a term's definition, the user positions the mouse cursor over the term and selects by depressing the first mouse button. The term is highlighted in reverse video and the definition is displayed in the viewing area. Subsequent selections from the display area cause the previous definition to be overlaid by newly selected ones.

If the user wishes to examine related terms, the control panel item labeled "Related Terms" may be selected. The Glossary prompts the user to enter a term for which related information is sought. If any related terms exist in the database, these will appear in the display area and may then be selected for definition.

By choosing the "Jump" option, the user may be positioned at an arbitrary location in the Glossary. This is a useful feature for moving around quickly when the Glossary contains many terms. The user is prompted to enter a term or letter of the alphabet. Once provided, the terms are listed beginning at this alphabetical location.

The status window keeps track of the user's interaction and provides terse instructional messages. At any time, the user may return to the Glossary main menu, collapse the Glossary into its icon representation, or exit.

The Glossary contents are updated and expanded through the Programmer's Assistance component of the Assistance Manager.

### *3.3.4 Local (Tool-Specific) Help*

Local Help refers to help available within a tool running in the SMDE. Such help is accessible in a uniform, consistent manner and meet the requirements of flexibility, expandability, unobtrusiveness, context-sensitivity, consistency, and maintainability. The Assistance Manager provides the means for this by giving tool programmers the ability to include a help package directly in their code.

Once inside an SMDE tool, the user may access help in any of three ways. By depressing the third mouse button on a help menu item, like the boxed question mark in Figure 5, the pop-up menu can be displayed from which "Intro to MDE", "Tutorial", "Glossary", and "Comments" can be selected. Once the selection is made, a new window appears and the Assistance Manager component runs in it independently of the tool which invoked it.

A user's interaction with a tool is typically driven by panel items which allow the user to select options and enter text. Help for these functions is enabled by positioning the mouse cursor over the panel item and by depressing the second mouse button. A small pop-up window appears with an explanatory or instructional message. (See the window on the top-left corner of Figure 5). The window remains visible until the user selects "Done".

Each tool window contains a small status area where error diagnostics and messages appear. A "Help-on-Tap" feature is provided via an "Explain" menu item which is displayed when an error message or prompt occurs. Selecting "Explain" results in detailed instructions or explanations of error conditions and why they may have occurred. The button may be selected repeatedly for progressively more detailed information. The button does not appear on the screen unless there is text displayed in the status window.

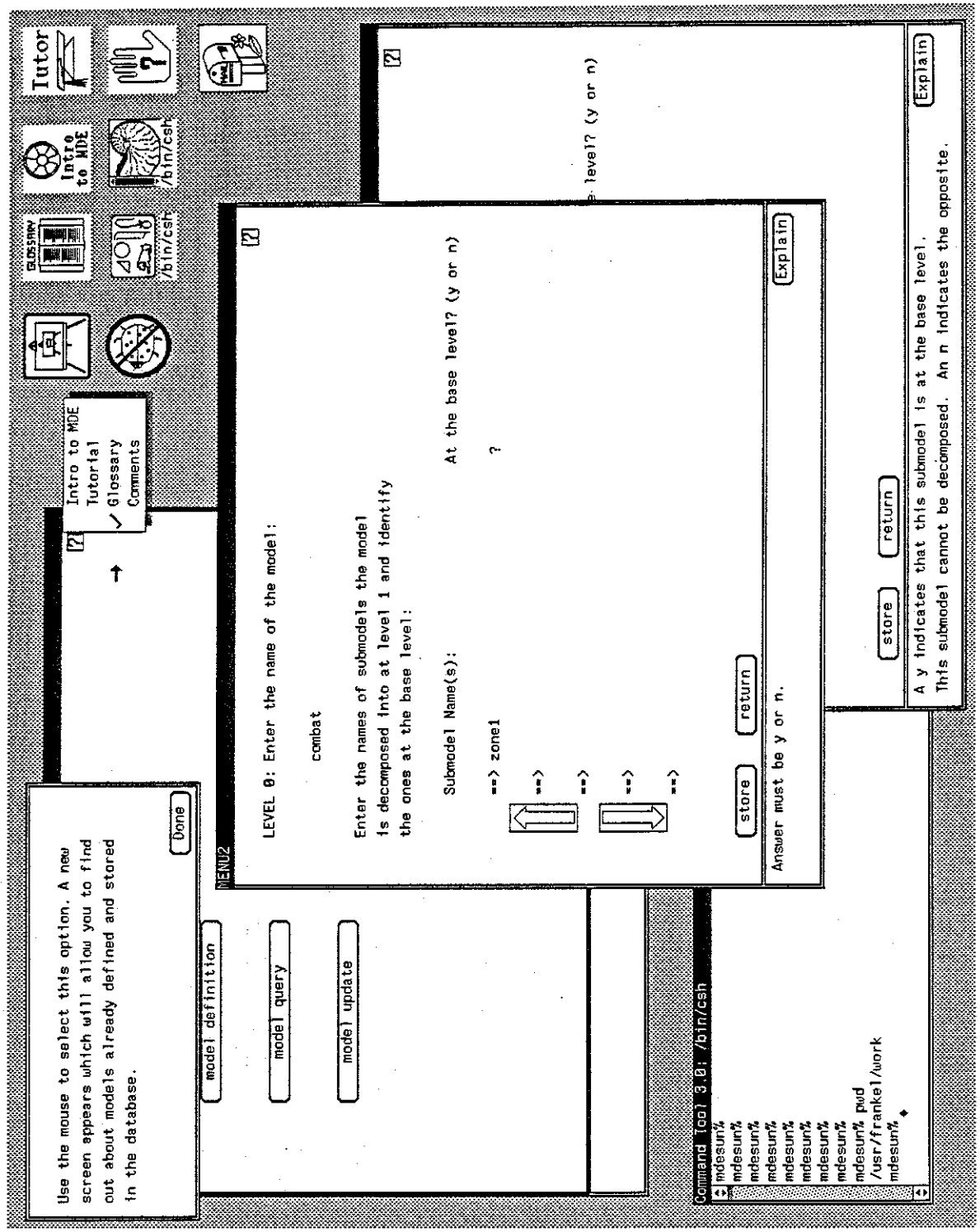


**Figure 5.** The *Local (Tool-Specific) Help* Component.

The features just described are achieved through a comprehensive set of modules which the tool programmer includes in his application code. Once the package is included, the programmer needs to make specific information available to the help package. Although this results in extra code for the application programmer, this code is nevertheless straightforward and easy to implement.

The text for the help messages and error explanations are prepared independently of the application program and must be stored in the help database. Apart from the composition of the help scripts, this task is automated and is further described in Section 3.3.6.

### *3.3.5 Comment Facility*

A comment option is included in the prototype to record user observations and suggestions (Figure 6). The comments are stored in the help database along with the user's name, date, and the tool identification. This facility provides a means to track user satisfaction with the Assistance Manager and is also useful for reporting "bugs."

### *3.3.6 Programmer Assistance*

Programmer Assistance deals with updating and maintaining the help database, and the use of the help package in an application program. Unlike the other components of the Assistance Manager, Programmer Assistance is available only to developers who have access authorization for its use. It is only available through the top-level Assistance Manager menu.

When Programmer Assistance is selected, a pop-up frame appears and prompts for an access code as shown in Figure 6. If an incorrect code is entered, the tool denies access and the frame disappears; otherwise, the top-level Programmer Assistance menu is displayed and the pop-up frame disappears. The options allow the user to add to or

**Figure 6.** The *Comment* and *Programmer Assistance* Components.

delete from the documents and tutorials databases, and to add, delete, and update the glossary definitions, diagnostic messages, and help scripts in the database.

Adding and deleting tutorials for the Tutorial and documents for the Introduction to the SMDE are handled similarly. Once the user selects an option, the Programmer Assistance window is overlaid by a new window which prompts the user for document name and location as shown in Figure 7. The documents are moved to the appropriate directory and the Table of Contents is updated.

Updating the glossary, diagnostic, and help script databases is accomplished by a direct interface to the database. The programmer may retrieve text for update or add new material at any time, although in most applications this would be done only after the tool developer had completed testing and debugging of his design. Invoking either the add or update option results in the replacement of the current screen by a new screen which is tailored for the feature chosen.

The "Reference" menu item on the Programmer Assistance window can be selected to view a Quick Reference Guide, shown in Figure 7, by way of scrolling up or down. This feature is provided as a quick lookup for syntax and usage of the high level procedure calls of the help package. A programmer who is developing an application for the SMDE can use the Reference as a memory refresher and avoid resorting to written documentation.

## 4. EVALUATION OF THE PROTOTYPE

The prototype has been evaluated with respect to the design objectives of Section 3.1. This evaluation is reported below for each design objective.

(1) *Provide general information for new users*: The *Introduction to SMDE* provides general information for new users or anyone else who is curious about the SMDE. This information is accessible in such a way that one may invoke it from the UNIX command level; therefore, one need not enter the environment to learn more about it.

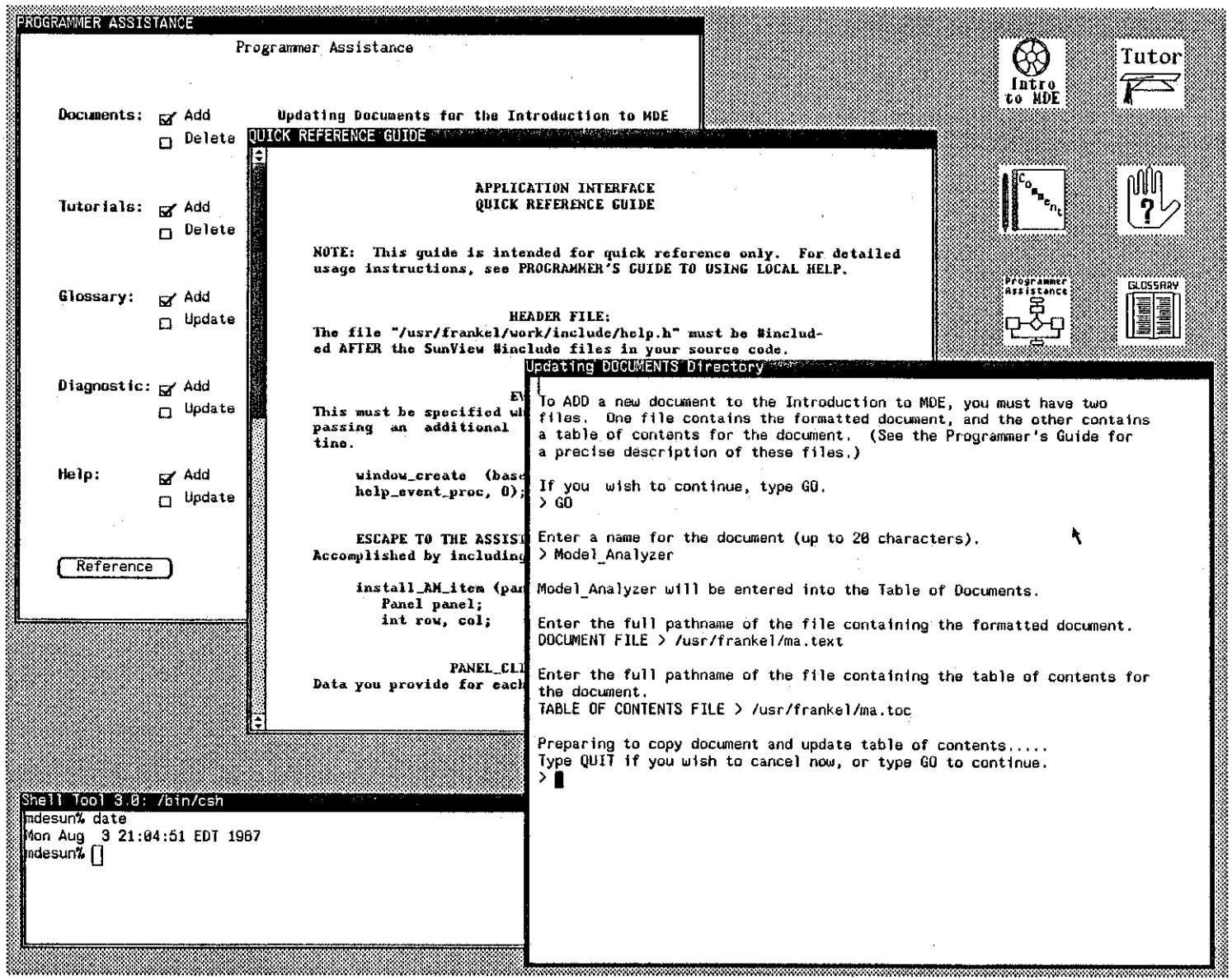


**Figure 7.** The *Programmer Assistance* Component.

(2) *Provide help for an SMDE tool*: One of the most important objectives of the Assistance Manager is satisfied by incorporating help access mechanisms directly with the tool. These mechanisms make up the *Local Help* component of the Assistance Manager.

(3) *Provide definitions of technical terms*: Definitions of technical terms can be found in the *Glossary* along with example usage and related information.

(4) *Provide tutorial assistance*: The *Tutorial* component of the Assistance Manager provides an interface to tutorials for potentially any tool or topic of interest.

(5) *Provide constantly available, immediately accessible help*: Help is always "active" in the sense that the user need only select a menu item to access it. The selectable panel items are an integral part of the SMDE user interface, and consequently appear on every tool screen. Their functionality has been expanded to let them be used to request help at any time.

(6) *Provide help that is unobtrusive*: The interface design has been done so that the user does not "see" the Assistance Manager until he "looks" for it. Help windows pop up and then disappear, and any Assistance Manager screen may be closed or moved to an unobtrusive location on the screen.

(7) *Provide help that is flexible enough for all levels of users*: Diagnostic messages and prompts can be requested in progressively greater detail at the discretion of the user. (It should be noted that although the Assistance Manager makes it straightforward for the tool designer and technical writer to give this flexibility, it is ultimately their responsibility to make use of this capability.) In addition, the Assistance Manager provides the *Tutorial* and *Introduction to SMDE* for general use, and provides more specific levels of detail in *Local Help* and the *Glossary*. The focus has been to accommodate the modeler who uses the environment frequently, providing a greater level of detail and general information only on request.

(8) *Provide context-sensitive help*: Context-sensitive help has been achieved wherever applicable. (Note that the focus of the *Introduction to SMDE* and the *Tutorial*, as well as the *Glossary* in most uses, preclude context-sensitivity as defined in the objectives.) The calls to the Assistance Manager through *Local Help* are "hard-wired" into the functions of the panel items themselves. It is always possible to determine the user's state and task in this implementation, allowing for a high degree of context sensitivity.

(9) *Provide appropriate access mechanisms*: Access to the Assistance Manager is designed around its intended use. When it is serving as a tutor or for general information, it is accessible from the UNIX command level or from a menu item on a tool screen. All components except *Local Help* may be accessed directly or sequentially by top-level menu. *Local Help*, as expected, is accessible only through a specific tool.

(10) *Provide systematic methods for tool programmers to build help into SMDE tools*: This objective is obtained by giving programmers access to a library of routines and definitions which are included as header files and function calls from within application code. (The functions and header files may only be accessed through the C Programming Language.)

(11) *Provide help that is available in a consistent manner from one tool to another*: The same library that meets Objective 10 ensures that access to the Assistance Manager database will be consistent across all applications. The centralized control maintained by the library guarantees uniformity in the user interface.

(12) *Serve as interface between user and the Assistance Manager database*: The Assistance Manager database is only accessible through Assistance Manager mechanisms. Updates and additions to the database can only be made after authorization has been granted by the system administrator. This is enforced by restricting database modifications to users who supply the appropriate access code once the Programmer Assistance tool has been invoked.

(13) *Provide easy methods for update and expansion of the Assistance Manager database*: All database updates may be made through the *Programmer Assistance* component of the Assistance Manager. Through a series of prompts and menus, authorization is verified, and the update proceeds through a controlled series of data entry "forms" — screens that are customized for a particular data entry operation. Data already present in the database may be edited or deleted, and new data may be added by the same conventions.

## 5. CONCLUSIONS

The research clearly indicates that most on-line assistance systems fall short of their potential despite the current emphasis on systems that are human-engineered. There has been little attempt on the part of implementors to build on the past successes of others; designers have consequently repeated many mistakes and been doomed to the same failures.

Today's software and hardware technology can inspire imaginative ways of integrating a sophisticated on-line assistance package with a software system. Online assistance should never have to be implemented in an ad-hoc or post-hoc fashion.

Perhaps the most important question remaining for software engineers is how to persuade the user to make use of on-line assistance. The truest test of a help system is whether it is used, and whether its use is an effective aid to productivity.

## ACKNOWLEDGMENTS

This research was sponsored in part by the Space and Naval Warfare Systems Command and Naval Research Laboratory under contract N60921-83-G-A165-B030 through the Systems Research Center at VPI&SU. The authors acknowledge stimulating discussions with R.E. Nance; R.L. Moose, Jr.; E.J. Derrick; and L.F. Barger which contributed to the development of the prototype.

## REFERENCES

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

Balci, O. and R.E. Nance (1987a), "Simulation Model Development Environments: A Research Prototype," *Journal of the Operational Research Society*, to appear.

Balci, O. and R.E. Nance (1987b), "Simulation Support: Prototyping the Automation-Based Paradigm," In *Proceedings of the 1987 Winter Simulation Conference* (Atlanta, Ga., Dec. 14-16). IEEE, Piscataway, N.J., to appear.

Bergman, H. and J. Keene-Moore (1985), "The Birth of a Help System," In *Proceedings of the 1985 ACM Annual Conference* (Denver, Colo., Oct. 14-16). ACM, New York, N.Y., pp. 289-295.

Blake, T. (1986), "The Art and Science of User Interface Design," Tutorial presented at the conference on *CHI '86 Human Factors in Computing Systems* (Boston, Mass., Apr. 14-17).

Borenstein, N.S. (1985), "The Design and Evaluation of Online Help," Ph.D. Thesis, Carnegie-Mellon University, Pittsburgh, Penn., Apr.

Bramwell, B. (1983), "BROWSE: An On-Line Manual and System Without an Acronym," *ACM SIGDOC Asterisk 9*, 4 (Apr.), 7-11.

Butler, T. and L. Kennedy (1985), "The AT&T Unix System Help Facility," *Unix/World 2*, 6 (June), 81-83, 102.

Card, S.K., W.K. English, and B.J. Burr (1978), "Evaluation of Mouse, Rate-Controlled Isometric Joystick, Step Keys, and Text Keys for Text Selection on a CRT," *Ergonomics 21*, 8 (Aug.), 602-613.

Chin, D.N. (1986), "User Modeling in UC, the Unix Consultant" In *Proceedings of CHI '86 Human Factors in Computing Systems* (Boston, Mass., Apr. 14-17). ACM, New York, N.Y., pp. 24-28.

Connolly, T., A. Bradford, R. Grice, and J. Steipp (1985), "Issues Concerning the Development and Use of Online Information," *ACM SIGDOC Asterisk 11*, 1 (Jan.), 21-30.

Estrin, G., R.S. Fenchel, R.R. Razouk, and M.K. Vernon (1986), "SARA (System Architects Apprentice): Modeling, Analysis, and Simulation Support for Design of Concurrent Systems," *IEEE Transactions on Software Engineering SE-12*, 8 (Aug.), 293-311.

Frankel, V.L. (1987), "A Prototype Assistance Manager for the Simulation Model Development Environment," M.S. Thesis, Department of Computer Science, Virginia Tech, Blacksburg, Va., July.

Gait, J. (1985), "An Aspect of Aesthetics in Human-Computer Communications: Pretty Windows," *IEEE Transactions on Software Engineering SE-11*, 8 (Aug.), 714-717.

Hodgson, G.M. and S.R. Ruth (1985), "The Use of Menus in the Design of On-Line Systems: A Retrospective View," *SIGCHI Bulletin 17*, 1 (Jan.), 16-21.

Houghton Jr., R.C. (1984), "Online Help Systems: A Conspectus," *Communications of the ACM 27*, 2 (Feb.), 126-133.

Kernighan, B.W. and R. Pike (1984), *The Unix Programming Environment*, Prentice-Hall, Englewood Cliffs, N.J.

Nakatani, L.H. (1983), "Soft Machines: A Philosophy of User-Computer Interface Design," In *Proceedings of the CHI '83 Human Factors in Computing Systems* (Boston, Mass., Dec. 12-15). ACM, New York, N.Y., pp. 19-23.

Nakatani, L.H., et al. (1986), "TNT: A Talking Tutor 'N Trainer for Teaching the Use of Interactive Computer Systems," In *Proceedings of the CHI '86 Human Factors in Computing Systems* (Boston, Mass., Apr. 14-17). ACM, New York, N.Y., pp. 29-34.

Norman, D.A. (1983), "Design Rules Based on Analyses of Human Error," *Communications of the ACM 26*, 4 (Apr.), 254-258.

Orwick, P., J.T. Jaynes, T.R. Barstow, and L.S. Bohn (1986), "DOMAIN/DELPHI: Retrieving Documents Online," In *Proceedings of the CHI '86 Human Factors in Computing Systems* (Boston, Mass., Apr. 14-17). ACM, New York, N.Y., pp. 114-121.

Relles, N., N.K. Sondheimer, and G.P. Ingargiola (1981), "A Unified Approach to Online Assistance," In *Proceedings of the 1981 National Computer Conference* (Arlington, Va.). AFIPS Press, pp. 383-388.

Robertson, G., D. McCracken, and A. Newell (1981), "The ZOG Approach to Man-Machine Communication," *Intl. Journal of Man-Machine Studies 14*, 461-488.

Roberts, R. (1970), "HELP: A Question Answering System," In *Proceedings of the 1970 Fall Joint Computer Conference* (Arlington, Va.). AFIPS Press, pp. 547-554.

Rothenberg, J. (1979), "Online Tutorials and Documentation for the SIGMA Message Service," In *Proceedings of AFIPS National Computer Conference 48*, pp. 863-867.

Smith, D.C., C. Irby, R. Kimball, and B. Verplank (1982), "Designing the STAR User Interface," *Byte*, (Apr.), 242-282.

Sondheimer, N.K. and N. Relles (1982), "Human Factors and User Assistance in Interactive Computing Systems: An Introduction," *IEEE Transactions on Systems, Man, and Cybernetics SMC-12*, 2 (Feb.), 102-106.

Stoddard, M., K. Berkbigler, B. Wheat, and E. Peter (1985), "User Behavior Upon Introduction of a Network Help System," *ACM SIGCHI Bulletin 16*, 3 (Mar.), 25-31.

Sun Microsystems (1986a), *SunView Programmer's Guide*, Sun Microsystems, Inc., Mountain View, Calif., Feb.

Sun Microsystems (1986b), *SunINGRES*, Volumes I, II, and III, Sun Microsystems, Inc., Mountain View, Calif., May.

Walker, J. (1986), "Online Documentation and HELP Systems," Tutorial presented at the conference on *CHI '86 Human Factors in Computing Systems* (Boston, Mass., Apr. 14-17).

Yestingsmeier, J. (1984), "Human Factors Considerations in Development of Interactive Software," *SIGCHI Bulletin 16*, 1 (Jan.), 24-27.

| REPORT DOCUMENTATION PAGE | | READ INSTRUCTIONS BEFORE COMPLETING FORM |
|---|---|---|
| 1. REPORT NUMBER<br>SRC 87-009 (TR-87-21 Dept. of CS) | 2. GOVT ACCESSION NO. | 3. RECIPIENT'S CATALOG NUMBER |
| 4. TITLE (and Subtitle)<br>A Prototype Assistance Manager for the Simulation Model Development Environment | | 5. TYPE OF REPORT & PERIOD COVERED<br>Interim<br>6. PERFORMING ORG. REPORT NUMBER<br>SRC 87-009 (TR-87-21 Dept. CS) |
| 7. AUTHOR(s)<br>Valerie L. Frankel and Osman Balci | | 8. CONTRACT OR GRANT NUMBER(s)<br>BOA N60921-83-G-A165 B030 |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS<br>Department of Computer Science<br>Virginia Tech<br>Blacksburg, VA 24061 | | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS |
| 11. CONTROLLING OFFICE NAME AND ADDRESS<br>Space and Naval Warfare Systems Command and Naval Research Laboratory<br>Washington, D.C. | | 12. REPORT DATE<br>1 August 1987<br>13. NUMBER OF PAGES<br>29 |
| 14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)<br>Naval Surface Weapons Center<br>Dahlgren, VA 22448 | | 15. SECURITY CLASS. (of this report)<br>Unclassified<br>15a. DECLASSIFICATION/DOWNGRADING SCHEDULE |

16. DISTRIBUTION STATEMENT (of this Report)

This report is distributed to Navy research and development centers and other university-based laboratories. A limited number of reports is distributed for peer review.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Simulation and modeling, design, human factors, model development environments, on-line assistance, simulation support systems.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)
The Assistance Manager, one of the tools of the Simulation Model Development Environment (SMDE), is required to provide effective and efficient transfer of assistance information to an SMDE user. This paper describes a prototype of the SMDE Assistance Manager. Objectives are set forth and a design is established and implemented on a SUN 3/160 workstation. The prototype is evaluated with respect to the design objectives and is shown to provide a highly flexible interface between the user and the database of assistance information. Results indicate that the prototype Assistance Manager incorporates the characteristics

considered desirable in on-line assistance systems and serves as a basis for future enhancement and development.