N93-19404

1992 NASA/ASEE SUMMER FACULTY FELLOWSHIP PROGRAM

JOHN F. KENNEDY SPACE CENTER
UNIVERSITY OF CENTRAL FLORIDA

DEVELOPMENT OF A TASK ANALYSIS TOOL TO FACILITATE USER INTERFACE DESIGN

PREPARED BY:	Dr. Jean C. Scholtz
ACADEMIC RANK:	Assistant Professor
UNIVERSITY AND DEPARTMENT:	Portland State University Computer Science Department
NASA/KSC	
DIVISION:	Shuttle Project Engineering Office
BRANCH:	Process Integration Branch
NASA COLLEAGUE:	Arthur E. Beller
DATE:	August 21, 1992
CONTRACT NUMBER:	University of Central Florida NASA-NGT-60002 Supplement: 8

Acknowledgments

I would like to thank Dr. Loren Anderson and Ms. Kari Stiles of the University of Central Florida and Ms. Carol Valdes of the Kennedy Space Center for their efforts in making the NASA/ASEE Summer Faculty Fellowship Program an enjoyable and educational summer. I would also like to thank the many NASA, Boeing and Lockheed personnel who provided answers to a wide variety of questions, demonstrated software systems and provided hardware and software support. The list of names is too long to include here but without the expertise of all this work could not have been accomplished.

I would especially like to thank all the employees in the Shuttle Project Engineering Office for making me feel so welcome during the summer. Having a summer professor in this area was a first but hopefully, will not be the last.

Abstract

A good user interface is one that facilitates the user in carrying out his task. Such interfaces are difficult and costly to produce. The most important aspect in producing a good interface is the ability to communicate to the software designers what the user's task is. The Task Analysis Tool is a system for cooperative task analysis and specification of the user interface requirements. This tool is intended to serve as a guide to development of initial prototypes for user feedback.

Summary

The user interface is an extremely important part of software. Computer users today are not, in general, computer experts but experts in other domains who are dependent on computer software to facilitate their tasks. Developing interfaces for these users is an expensive and time consuming task. It is often difficult for the software developers to understand the user's domain well enough to come up with a usable interface. An iterative design process based on the concept of prototyping is becoming popular today. In this methodology a rapidly developed version of the software is used to obtain user feedback. This version lacks much of the eventual functionality and is used mainly to test out ideas the designers have about how the user interface should look. While the use of prototyping has proven to be valuable in the production of good interfaces, designers are still faced with the problem of developing initial prototypes and incorporating user feedback into the design of the interface.

This work presents a tool to be used in cooperative task analysis. End users and human-computer interaction personnel work together with the Task Analysis Tool to produce a task analysis and a rough sketch of an interface to support these tasks. The tool holds promise as a communication medium between end users and software designers. Better communication means fewer iterations in the interface design while still producing more usable interfaces.

Table of Contents

List of Figures

I. Introduction

1.1 The Design of User Interfaces

An important consideration in software development today is the interaction of the user with the software. This concern has emerged due to the changing nature of users of computer systems and the increasing complexity of current software systems. Today's users are not restricted to "computer hackers" ; they are, in fact, using software systems merely as a tool to aid in different aspects of their jobs. Therefore, the amount of time users have to devote to learning and using the system is limited, as is the amount of frustration they will tolerate. To add to this problem software systems are becoming increasingly complex. This presents a problem for both users and developers. Users often have a difficult time in accessing all the desired functionality. As the interface is essentially the view that the user has of a system, he must be able to clearly see through this interface to the functionality of the software (Shackel, 1988). Instead many of the systems today present a bewildering array of choices for the user. Developers are also faced with maintaining and augmenting complex code. The end result is that dealing with the software either as a user or developer requires a large amount of time and hence is a costly effort.

In order to address these problems an iterative process of software development is stressed. The underlying principle is that changes to the software are easier and less costly to make early in the development cycle. Prototyping is one way of collecting information from the user about the usability of the system early in the software design process (Wilson and Rosenberg, 1988). The user's view of a given software system is determined largely by the interface to that system. That is, system functionality that is not readily accessible in the interface is nonexistent as far as the user is concerned. The software interface should provide a good match with the task that the user must perform with the software. A prototype of the interface is often used to collect users' reactions and feedback to such things as terminology and arrangement of menu items, format of information presented and sequence of movement. This information is then quickly incorporated into the prototype and more user feedback is collected.

Boker and Gronbaek (1991) have studied the use of cooperative prototyping. They contrast this approach to one where designers develop prototypes on their own using information supplied by users. They view cooperative prototyping as a way to overcome problems in developing applications that more closely match user tasks. Initial prototypes are used to make the views of the participants concrete. Prototypes can be refined or replaced as users and designers actively participate in the design process. HCI (human-computer interaction) personnel in this approach need to become familiar with the tasks of the users. Initial prototypes are set up by the designers based on their understanding of the user's tasks. The authors found that both well constructed prototypes which display sample user data and mock-ups which allowed more flexibility in interaction were helpful in obtaining feedback. This approach still relies on an iterative method with the designers having the responsibility for construction of the initial prototype.

1.2 Obstacles to Iterative Design

This iterative procedure results in an interface that the user is pleased with and in timely feedback to the developers. There are, however, several obstacles to an efficient use of such a procedure in the real world. In many instances, software is developed on a contractual basis. This means that the product is agreed upon prior to any design. This agreement usually takes the form of written requirements based mainly on the functionality which the software is to provide. Specifications for the user interface usually do not exist, or if they do, they are merely platform and style specifications. In addition, the requirements are usually generated at the management level. The management level on the developmental side agrees to these. The actual software developers and the actual end users may or may not have participated in this interaction. Therefore, the interface produced often differs drastically from what the users may have expected.

Changes in design are difficult to make in this type of environment. Developers are often removed from the end users both organizationally and physically. Time constraints often make it difficult for the users to schedule large blocks of time or a series of sessions to work with the developers. Therefore, there is little chance for iterative development. Even when iteration exists, the necessary changes may not be incorporated due to the contractual agreement.

Large product development organizations also contain obstacles to user involvement as documented by Grudin (1991). Product development organizations are companies that develop and sell interactive software applications. The development process is separated into two parts: events prior to the start of the project and events during development. Although the time line that separates these processes is difficult to define, budgets and personnel are allocated according to these distinctions. The high level product description used in the early stage generally does not include the user interface despite the fact that it is difficult to draw a line between functionality and the interface. User involvement and interface issues are, therefore, issues that are addressed during development.

Moving some of this involvement to the design phase is a goal of HCI personnel. In Grudin's study rapid prototyping was found to be a useful tool in facilitating cooperative design. Moreover, the need to communicate information about computer use of the user directly to the developer was identified. Therefore, tools and methodologies that can be used to move user involvement to an earlier phase in the software development process are needed. Methods for developing and communicating user interface specifications to software developers are also greatly needed. The work presented here discusses a tool to accomplish this. This tool captures task analysis information directly from the end user and develops a rough initial prototype of the interface.

II. Task Analysis

2.1 Description of Task Analysis

Task analysis is a methodology for describing and analyzing performance demands made on the human element of a system. The goal of task analysis is a total human-machine system consisting of human performance requirements, hardware performance requirements and software performance requirements. Hardware and software requirements are much easier to obtain than are the human performance requirements and their interactions with the rest of the system.

The main objective of task analysis is to explore the relationships between the user's performance and the properties of the system. The focus is on designing a user interface to a system which is efficient and compatible with the view the user has of task performance. Design of dialogs in the interface is also a branch of task analysis. Maddix (1990) states that much dialog is based on an incomplete understanding of what kinds of interaction might take place between a typical user and the system. In doing task analysis the user's interaction with a given system is viewed with respect to the objects in the system and operations that the user performs on those objects. States in the system are changed by performing a sequence of operations on a series of objects. A goal can be described as a certain state within the system. This goal can be achieved by applying sequences of operations to objects in a given state. Guindon (1988) identifies these steps in task analysis:

1. Identify objects
2. Identify operations
3. Identify the sequence of operations used

The human constituents of a system are responsible for recognizing and interpreting states produced by the hardware and software systems. If these states have not produced the desired goals then it is necessary for the human to interact with the software to produce this state. Human error in carrying out these functions can not be completely eliminated but providing systems that are well matched to the users' tasks help in reducing the margin for error.

Don Norman(1986) identifies the gulf of execution and the gulf of evaluation in human-computer interaction. The gulf of execution results when the user is unable to correctly select the necessary sequence of operations to perform in order to produce the desired goal. The gulf of evaluation results from an incorrect interpretation or recognition of the state produced by a sequence of operations. The user bases this interpretation on the feedback produced by the system. The gulf is created by the difference between the user's view of what is happening and what is actually happening in the system. This distance is reduced as the user's view more closely matches the system view. Therefore, the interface and, consequently the dialog, between the user and the system must be the vehicle that maps the user's task into the functional components provided by the software.

A task analysis can be used to provide data about the user component of the system. The major problem then becomes how to map this task analysis data into an interface description that can be used to guide software developers in system design and

implementation. The data produced by a task analysis can take many forms depending upon the purpose for which that data was collected. In the case of this work, the concern is with the user interface so the task analysis will focus on interface items such as data displayed, format of the data, actions or operations on that data and the sequence in which the tasks are performed.

2.2 Obstacles In Performing Task Analysis

In order to produce a task analysis human-computer interaction personnel need to observe the user carrying out the task and to identify the objects, operations and sequences used. Additionally, in carrying out a task analysis for development of a software system, one must keep in mind that the current task will be changed by this automation. This means that the present task analysis must be examined to ascertain the effects that automation will have or that flexibility will have to be built into the system to accommodate future changes in task performance.

Many tasks involve a cognitive aspect. Users choose objects and operation in the system based on domain knowledge. In order to produce an effective interface it is necessary to understand these decisions. As the domains become increasingly complex this presents a larger obstacle to carrying out representative task analyses. Either the human-computer interaction person needs to learn the domain or the domain experts need to learn how to do task analysis.

In addition, domain experts often have difficulty in explicitly stating portions of their task. Portions of any expert's job become routine after a period of time and these routine cognitive tasks become difficult to verbalize. The human-computer interaction personnel is therefore responsible for recognizing missing portions and probing further to extract this knowledge from the expert. This puts additional demand on HCI personnel to understand the domain.

III. The Task Analysis Tool

3.1 Objective of the Task Analysis Tool

The main objective in the development of a tool to use in task analysis is to facilitate communication between the domain expert, HCI personnel , and the software designer. The following is a quote from Walsh, Lim and Long (1988):

> *"Human factors engineers complain that their contribution to iterative systems design is typically sought late, that is following system implementation. Software engineers, in contrast, complain that the human factors contributions to system design are neither timely, appropriate nor implementable. "*

The Task Analysis Tool (TAT) is designed to be used interactively by the domain expert under HCI supervision. Data collected during an interactive session will be analyzed by HCI personnel and given to the software designer to user as a guide to design of the interface. The data collected is saved in two forms: textual information that can be analyzed later for consistency issues within and between interfaces. Additionally, and more importantly for the user, a rough sketch of the interface is generated as information is entered. These screens can then be played back by the end user to help ensure that the displays give complete information in order to accomplish the given task. The Task Analysis Tool can serve as a useful tool to help the end user form a concrete description of his task. As visual feedback is provided immediately the user can match these results to his conceptual model. Corrections can be made to the interface sketch if the user finds that it is incorrect.

The fact that a rough sketch of the interface is produced serves to give a version to the user that is easier to survey for completeness than lists of functional requirements. The rough sketch can be used, in addition to functional requirements, to drive design. Having this sort of information at an early stage of design should mean that a better prototype can be initially developed. This serves in cutting down on the number of iterations that will be needed in obtaining user information. When given to the software developer this rough interface design serves to illustrate the control flow that the user follows. The task analysis tool is also a vehicle for agreement of expectations between users and developers.

3.2 Information Collection in the Task Analysis Tool

There are many definitions of tasks but a general agreement is that a task is composed of a set of human actions that contribute to some objective and ultimately to the output goal of a system. The content of a task can be more specifically defined once the objective of a task is identified.

Drury, Paramore, Van Cott, Grey and Corlett (1987) give the following characteristics useful in defining tasks :

"1. Task actions are related to each other not only by their objective but also by their occurrence in time. One of the concerns of task analysis is to establish and evaluate the time distribution of actions within and across tasks. Task actions include perceptions, discriminations, decisions, control actions, and communications. Every task involves some combination of these different types of cognitive and physical actions.
2. Each task has a starting point that can be identified as a stimulus or cue for task initiation. A cue is often not a single item of data or information. It may consist of several data points, received closely in time or dispersed over a longer time, which together have significance as a cue that an action is to be taken.
3. Each task has a stopping point that occurs when information or feedback is received that the objective of the task has been accomplished.
4. Task cues and feedback may be provided by instrumentation or direct sensory perception, or they may be generated administratively, say, by a supervisor or co-worker.
5. A task is usually, but not necessarily, defined as a unit of action performed by one individual."

The Task Analysis tool captures much of the information deemed characteristic of tasks. Some of this information is included in the sketched interface while other portions of it are included in the data file produced.

Tasks are of three types: discrete or procedural, continuous or tracking, and branching. Discrete tasks require that a series of actions be executed in response to a stimuli or procedure element. A continuous task extends over a long period of time, often cycling through a series of actions. A branching task is determined by the outcome of a certain action within the task.

The prototyped version of the Task Analysis Tool is most useful for discrete or procedural task and continuous tasks. Branching tasks cannot currently be handled but the addition of multiple path links will support this. The example presented in this paper contains an instance of a branching task. Therefore, a link that currently exists in the example might, in reality, not appear.

3.3 Status of the Task Analysis Tool

The Task Analysis Tool (TAT) currently exists in a prototyped version. While there are many features that have already been identified for addition into the system, this prototype should illustrate the usefulness of such a tool. Features suggested for inclusion in a developed version are discussed in section V.

TAT is designed to operate as two side by side displays. The screens that collect information constitute one set of displays. These are presented beside of the interface being sketched. The prototype of TAT was created using Toolbook by Asymetrix (1989) under Windows 3.1.

The figures that are included in this paper were printed using the print facilities of Toolbook. Unfortunately, menu bars do not print out. Therefore these have been separately constructed and included in figures where needed to illustrate the functionality of TAT. The operable version of TAT , therefore, looks slightly different than the version depicted here. In addition, Toolbook does not include the capability to print out dialog boxes. The example indicates which buttons are dialog boxes and describes the choices that are presented. In addition, the sizes of the displays have been adjusted slightly in the printed version in order to accommodate difference in the type size displayed on the screen and the printed type size.

A data file produced from a session with TAT contains information about the task, actions, information displayed and sequencing. Ideally this data could be examined to help in designing interfaces that will accommodate several viewpoints. Moreover, if several applications are to be used concurrently, examination of the data could be used to suggest commodities that should be considered in designing a common user interface. The current file produced is a labeled ASCII file. An example of this data file is included in Figures 14 and 20 and discussed more in section IV.

3.4 Description of the Task Analysis Tool

The Task Analysis Tool asks the user to identify the different tasks used to carry out a given process. These tasks are input as menu items in the constructed interface. A display is created for each discrete task. Figure 1 shows the initial display that is used to collect information about the names and numbers of tasks that makeup the process. Currently, the prototyped version of TAT allows only eight tasks per process. Figure 2 shows the complete set of forms that would be displayed if the user had indicated that eight tasks constituted this particular process. The user is asked to enter the tasks in some sort of meaningful order - either sequentially or in order of frequency of use. This is due to the order in which the items are entered into the menu of the sketched interface.

For each task, a display is presented (see Figure 3) that collects the information to be viewed and the type and format of the information. A task allows for two primary information sources, two secondary information sources and four status indicators. Each task can have up to six actions that are carried out on the information displayed. These actions are presented as sub menu items under the task menu item. The information that the users specifies is to be on the display is roughed out and presented according to the importance (primary, secondary or status) that the user assigns to the information.

Control flow in the system is represented by users specifying the tasks that precede and succeed the current task. These tasks are presented to the user in a dialog box that is constructed using the task names initially input on the first display in TAT. In the interface begin generated the displays for these tasks will be linked in the corresponding fashion so that the user can later play this back to assess if the flow accommodates the process correctly.

Figure 4 contains a template on which each interface display is sketched. For each new task in the process, this page is filled in with information collected from the user.

Figures 5, 6, and 7 are examples of help screens that are provided with TAT. The prototyped version contains no error handling capabilities. Although error detection and more help information will be included in the actual implementation of the software,

more help information will be included in the actual implementation of the software, TAT is intended to be used under the guidance of HCI personnel.

Figure 8 shows the screen that TAT displays when the user has finished typing in the information. At this point in time, the user can exit the system or can run the sketched interface to determine its correctness. If the user chooses to exist at this point, the interface will be saved and can be run later. The name of the data file selected by the user will also be displayed on this screen.

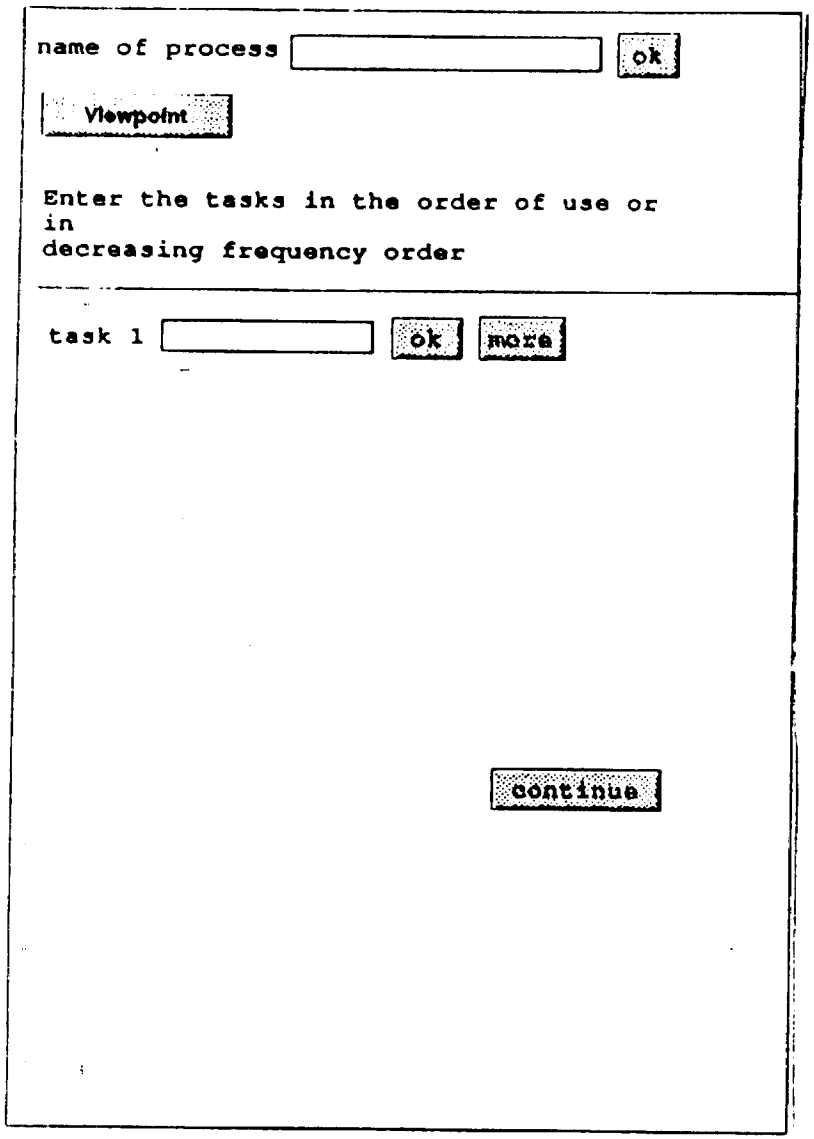

Figure 1: Initial Display of TAT

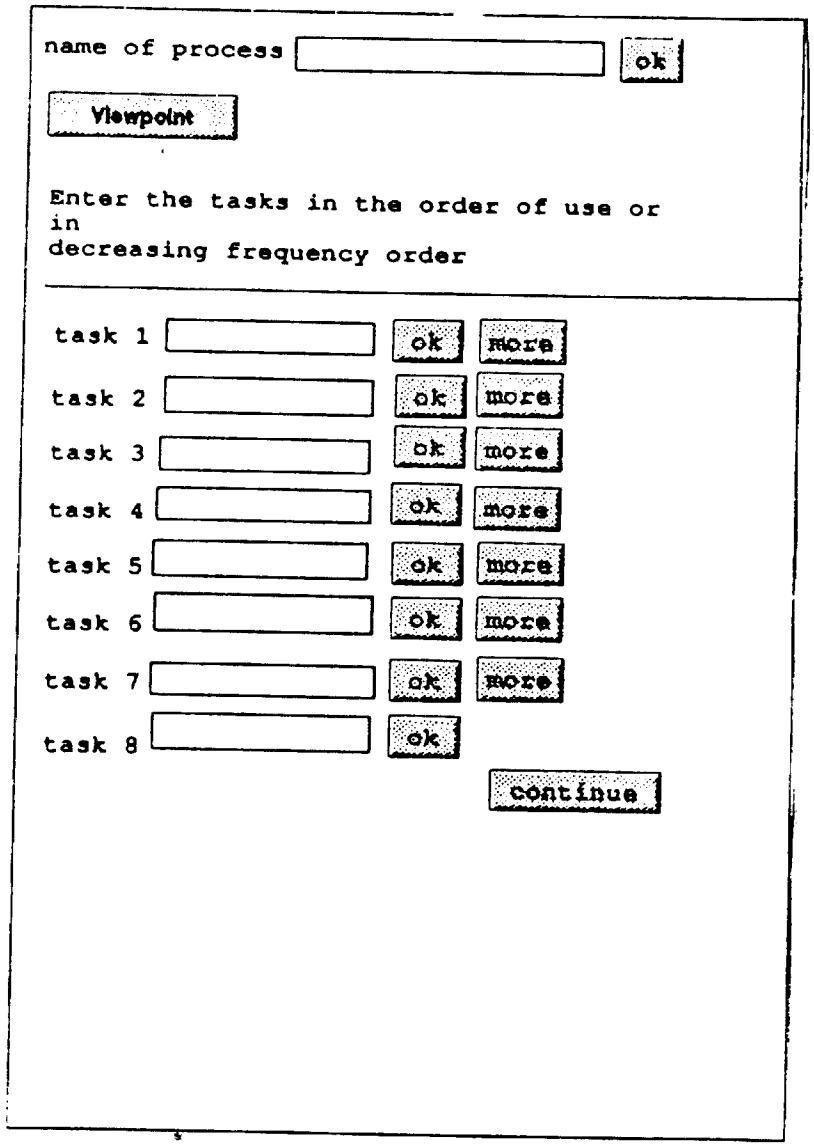

Figure 2: Initial Display of TAT Fully Expanded

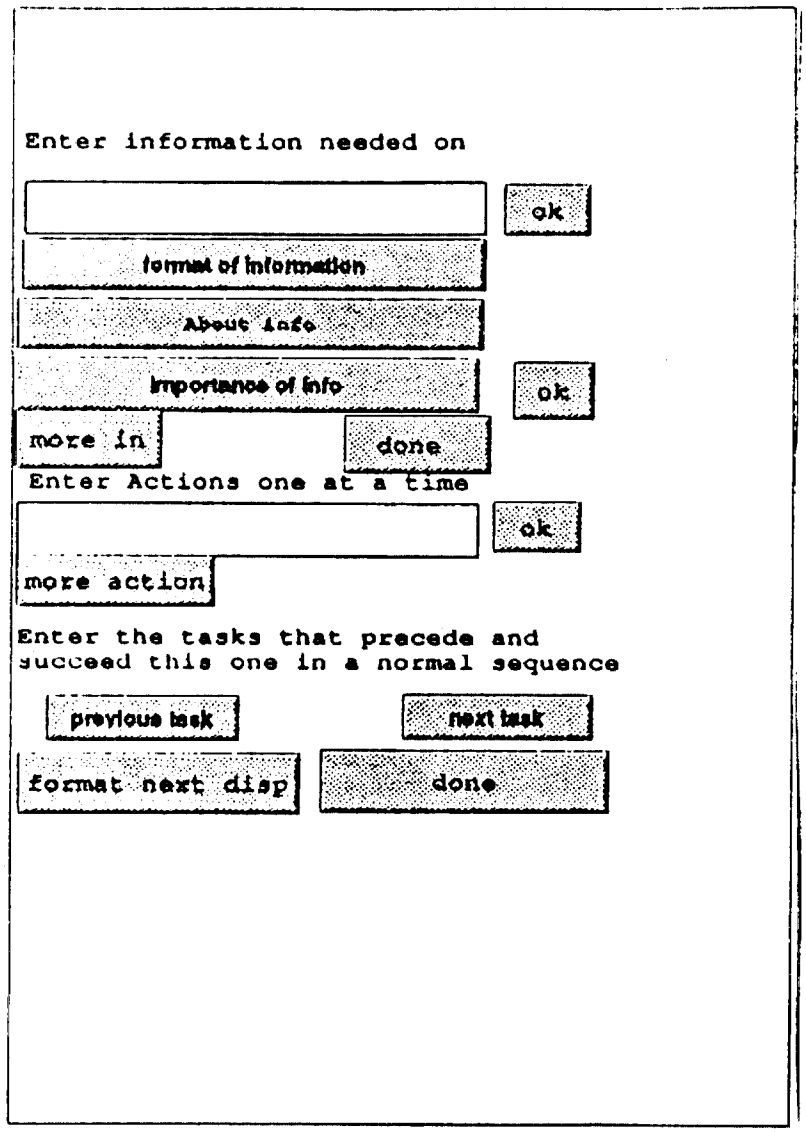

Figure 3: Information Collection Display of TAT

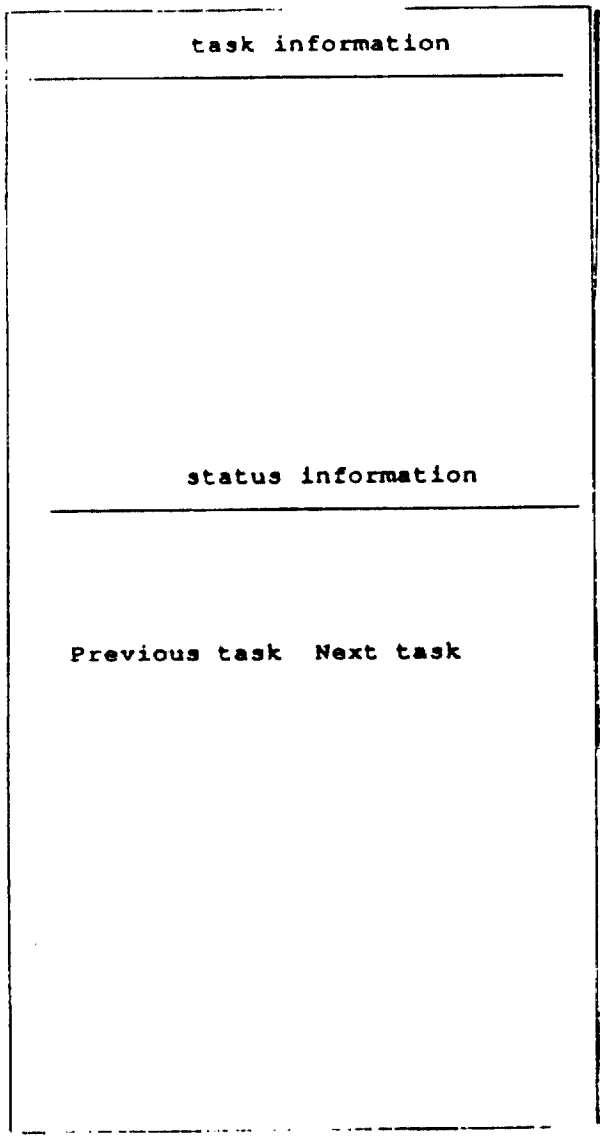

Figure 4: Template for Interface Sketch

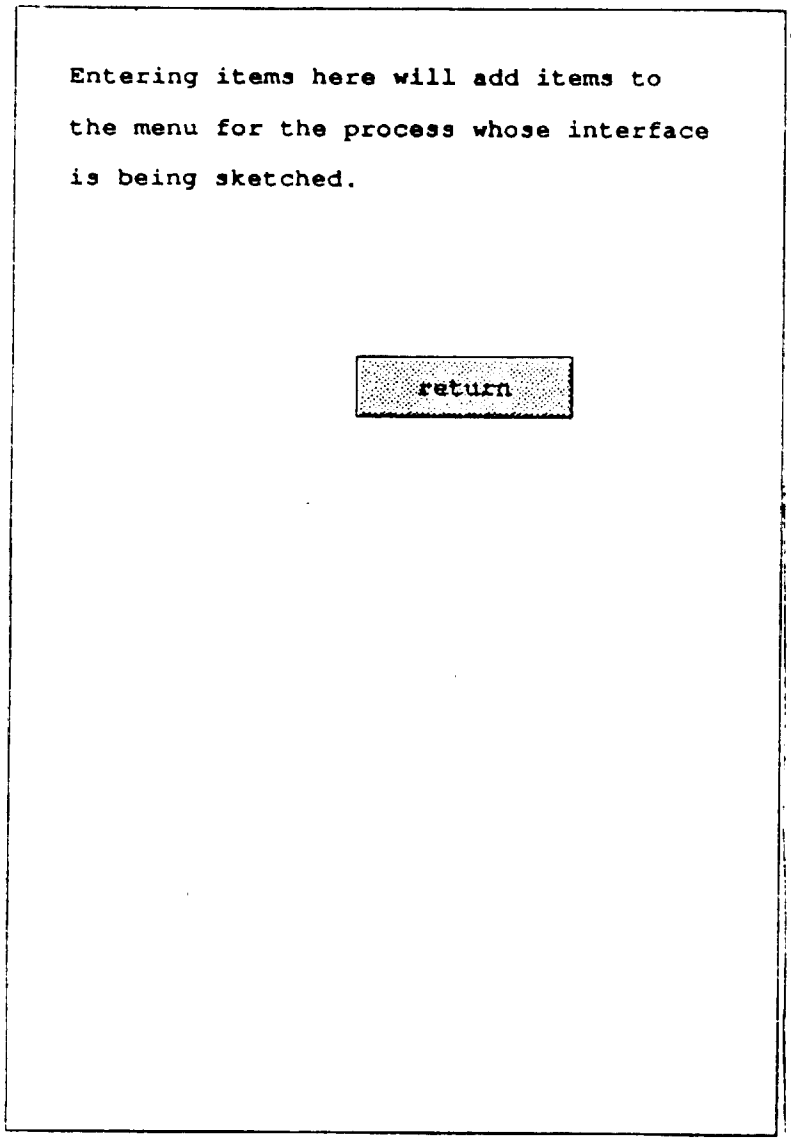

Figure 5: TAT Help Screen

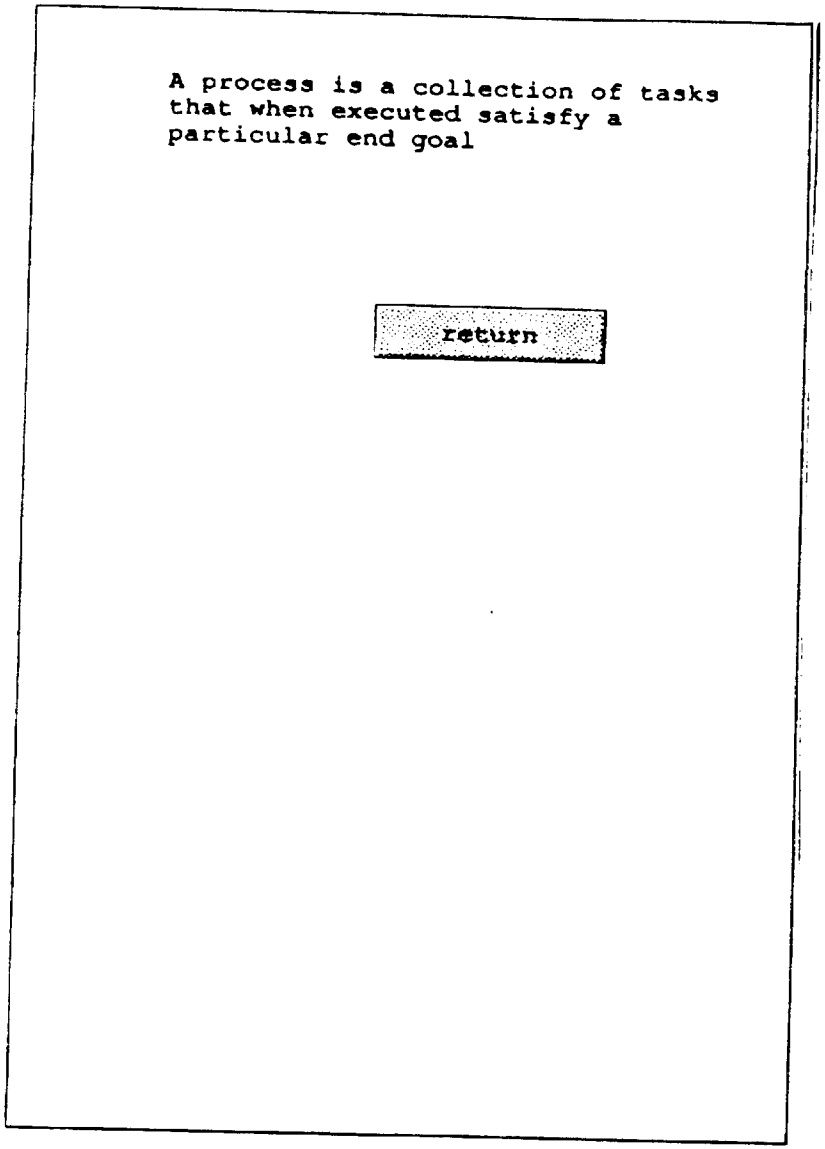

Figure 6: Another TAT Help Screen

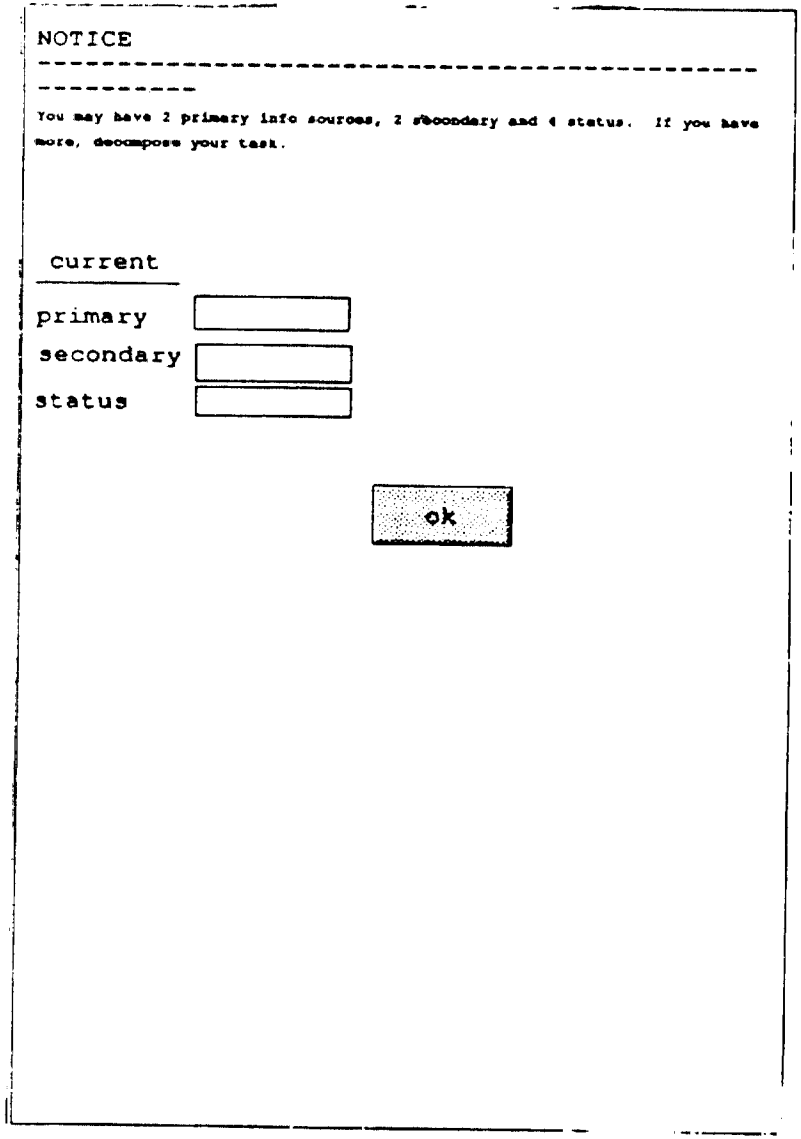

Figure 7: TAT Display for Information Types

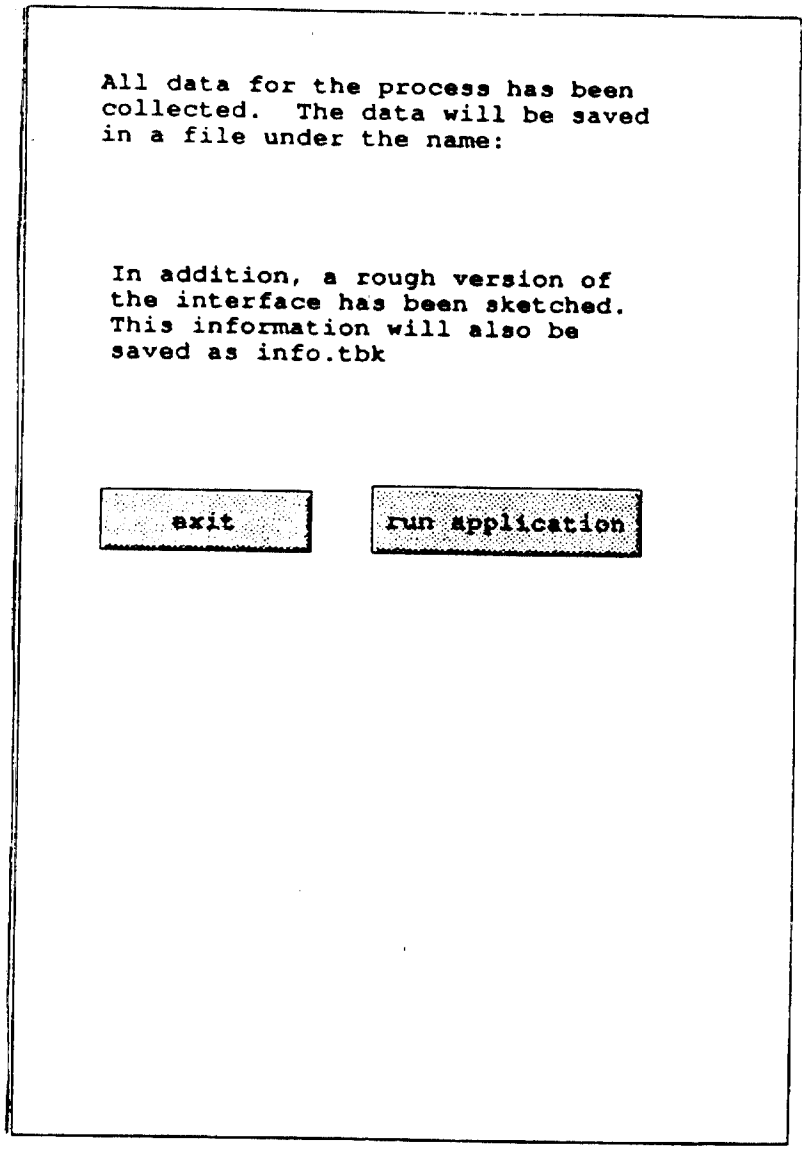

Figure 8: TAT End Display

IV. Example of the Use of TAT

4.1 Description of the Example Task of Electronic Review and Approval

The example presented here is an interface sketch for doing electronic review and approval. The following sections discuss the electronic review and approval process, the example, and an informal collection session using an early paper version of TAT.

In order to make changes to processes in the Shuttle Processing Environment at KSC, change requests must be generated and approved by the systems that are affected. Changes are done at various times before and during any given flow and range in size from large volumes of documentation to changes to a single operation or a change in sequence. A revision to a document is generated by selecting the portions of a master document that are to be used in this particular shuttle flow. Revisions include any changes that were generated previously to operations included in this flow. After a revision has been produced, changes that are made are termed deviations. A deviation may be a change in sequencing or a change to an individual step or steps. Deviations may be temporary. That is, the change is made only for this flow. A permanent deviation means that the change should be incorporated into this operation for all succeeding revisions. Currently these changes (revisions and deviations) are generated by engineering personnel and distributed to NASA personnel and other engineering teams for review and approval.

The review and approval process consists of suggesting changes to the text if necessary, making comments as appropriate or approving the change. During this process a reviewer may wish to see the comments or changes that other reviewers have generated. This procedure is an iterative one as comments and changes may need to be incorporated into the change and the change again distributed to the reviewers.

A computer based version could speed up the process. Reviewers could be notified electronically that a change was ready for review. The individual or group who initiated the change would be able to distribute it to the reviewers without having to either mail or hand deliver hard copies to the various individuals. Comment and changes made by the reviewers would be sent back electronically and could be directly incorporated into the change description. Reviewers would be able to quickly view other comments and the status of the change could be tracked electronically.

4.2 Example of the Process used to Sketch an Interface

The TAT example presented here uses the electronic review and approval process. Two interfaces are sketched here. These interfaces are from two different viewpoints: from the view of an engineering generating the change and from the point of view of a NASA reviewer. The software interface generated must support both views. Soliciting information from both viewpoints will yield data on commonalties that exist and the different emphasis that exists for the different views.

The first viewpoint presented is that of the reviewer. Figure 9 contains the initial information collection display. The name of the process input by the user is used to name the data file that will contain the information collected. The three tasks in the review and approval process are : select, review and approval. As these tasks are

entered by the user, these names appear in the menu bar of the sketched display. This is illustrated in the menu bar of the template for the interface sketch. In addition, a blank display is created on which the interface for each will be sketched. The viewpoint button is a dialog box which queries the user as to his position. In this case, the choices are : engineer, NASA, quality, NASA Test Director, documentation, and other.

Figure 10 contains a portion of the information collected for the task review. Figure 11 contains the interface generated by TAT that corresponds to the data entered in Figure 10. The first field on the display in figure 10 collects of the name of the various sources of information needed on the display. The format of information button is a dialog button that queries the user as to the way information is presented. The choices currently display in this dialog box are graphical, text, labeled data, tabular data and schematic data. The about info and importance of info buttons are a pair. The importance of info is a dialog box that asks whether the user considers a particular piece of information to be of primary or secondary to the task at hand. A third option would be to view the information as status only. TAT contains parameters which limit the number of primary, secondary and status pieces of information that can be concurrently displayed. The limitations in place now are rather arbitrary. In any given domain better limitations could be selected depending on display size , screen resolution and frequency and duration of use. The about info button is linked to a display (see Figure 7) that keeps track of the number of different information types presently selected for a given display. When the importance of information choice has been made and the OK button pressed, a labeled box will be drawn on the interface sketched. The size of the box will differ depending on whether the importance is secondary or primary. The box is also labeled as to ;how the data will be presented. Status only information appears as a button. If the user wished to enter more information sources, pressing the more info button clears the text field and focuses the cursor there.

When all the information sources have been entered, the user is prompted to enter actions that will be performed. These actions are entered as sub menu items. In order to see these, the user must highlight the task menu item in the sketched interface. The previous task and the next task buttons are dialog buttons that present the user with the list of tasks he has identified as being in this process. In the case of previous task, the list is augmented with start and in the case of next task the list will also contain end. Selecting a choice from these dialog boxes will result in a previous task and next task button being drawn on the interface sketch and in those buttons being linked to the correct display. This facilitates running the application. By pressing the previous or next task button, the user can simulate stepping through the process.

Pressing the format next display button brings up a blank information collection screen for the next task (in the order originally entered by the user on the first display) and a blank template for the interface sketch. After the information has been filled in for all the tasks in the process, the users can select the done button. Several things will then happen. First, information collected will be written to the specified data file. Then the user is asked whether he wishes to run the application just created. If he chooses not to, he can always retrieve this later from the "info.tbk" file and execute it.

Figures 12 and 13 are additional displays generated for the tasks of selecting items to review and approving or rejecting the changes. Figure 14 contains a portion of

the data file that was generated during this session. It contains information about the tasks, the information sources, the actions and the sequence of flow.

Figure 15 shows the initial display that was used to collect the tasks from the engineer's viewpoint. Notice that a task labeled create now exists and is used to initiate the change. In addition a release task exists where notification about an approved change and its incorporate into the master file is accomplished. Figure 16 is the display generated for the task of creating a change.

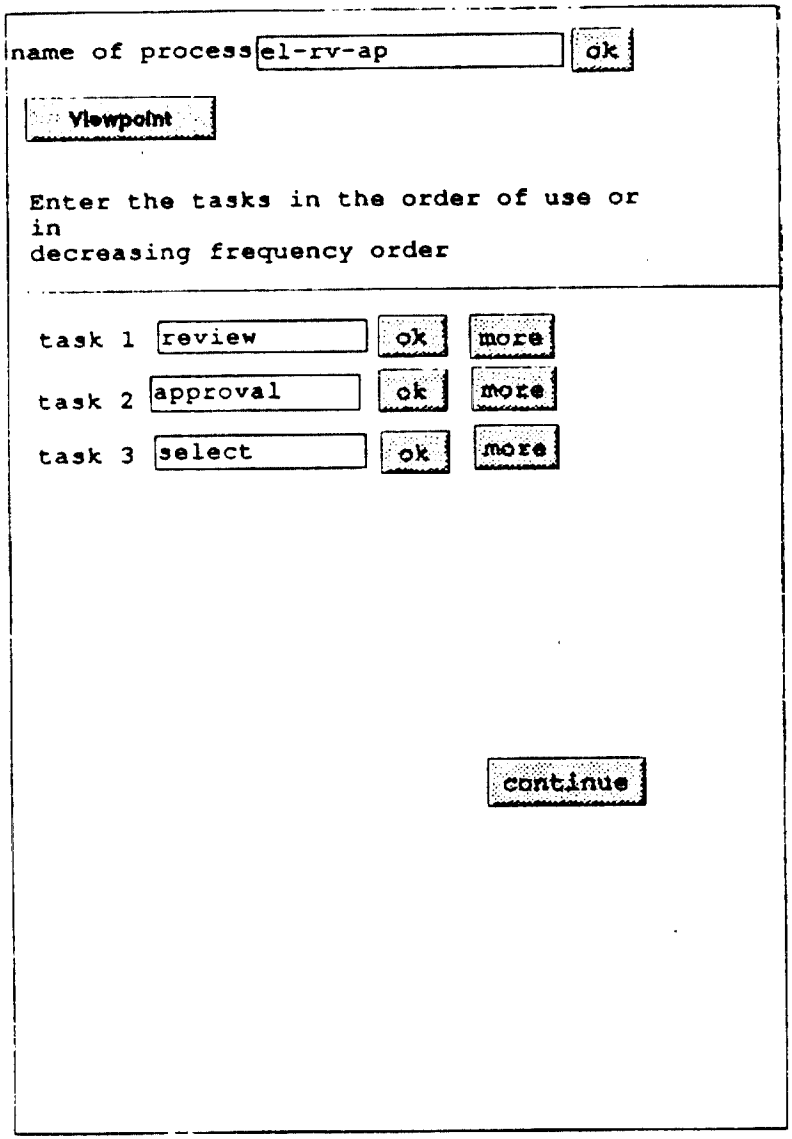

Figure 9: TAT Display for Example and Blank Template for Interface Sketch

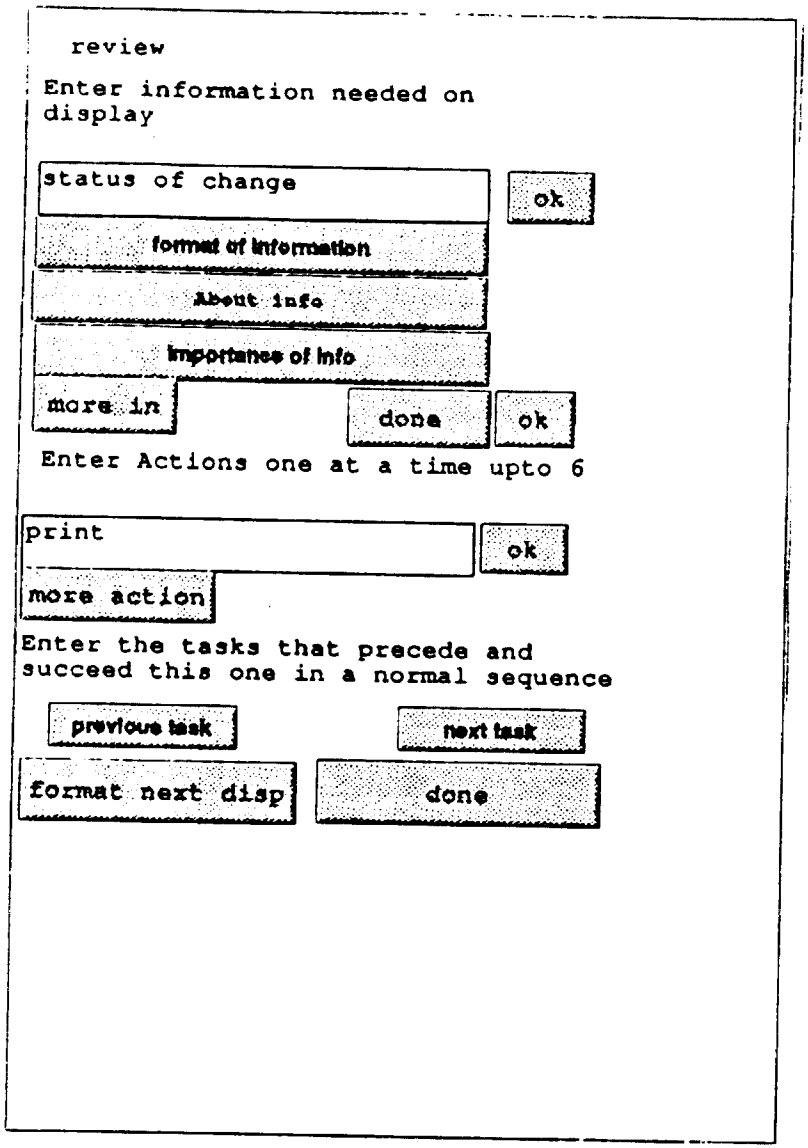

Figure 10: Information Collection Display from TAT for Task "review"

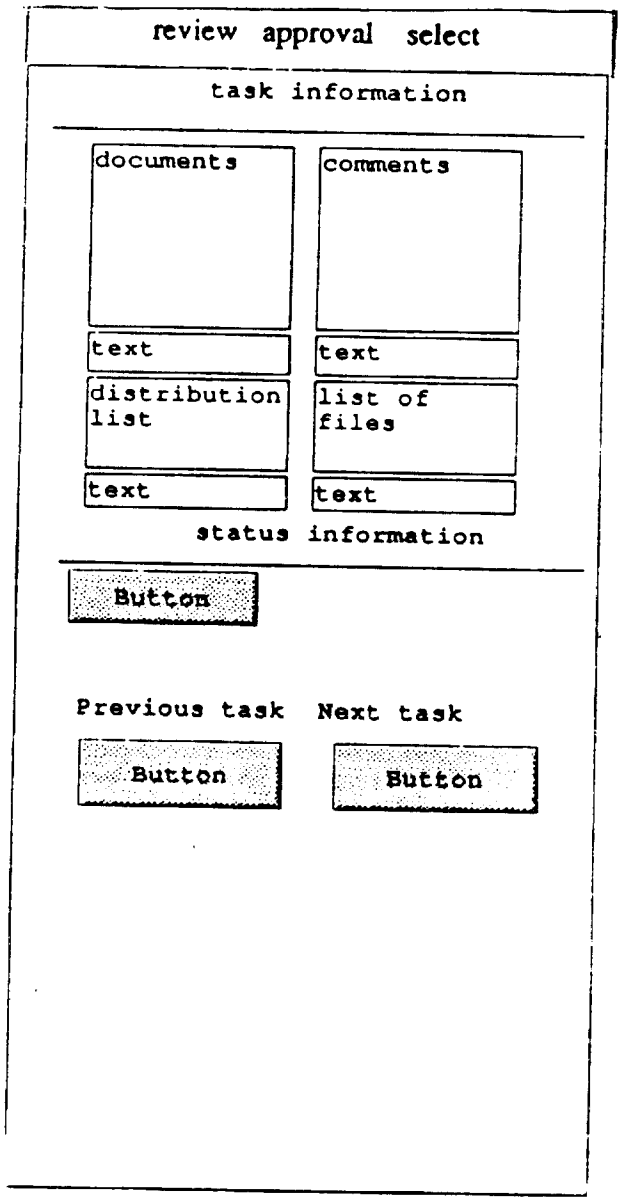

Figure 11: Sketch of Display Generated for "review"

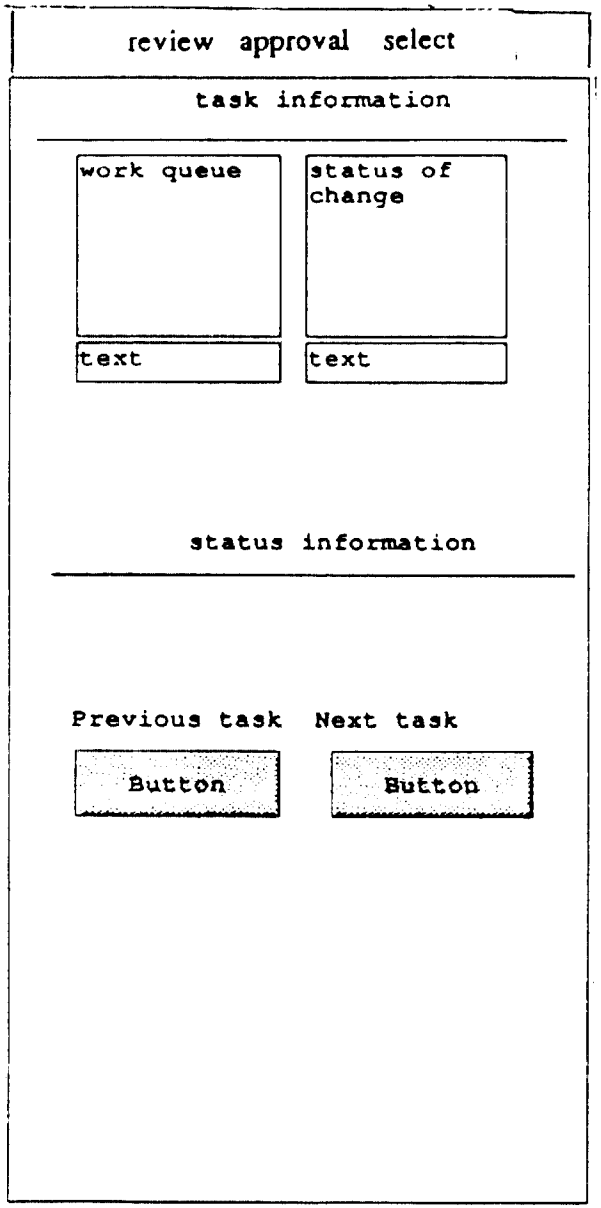

Figure 12: Sketch of Display Generated for "select"

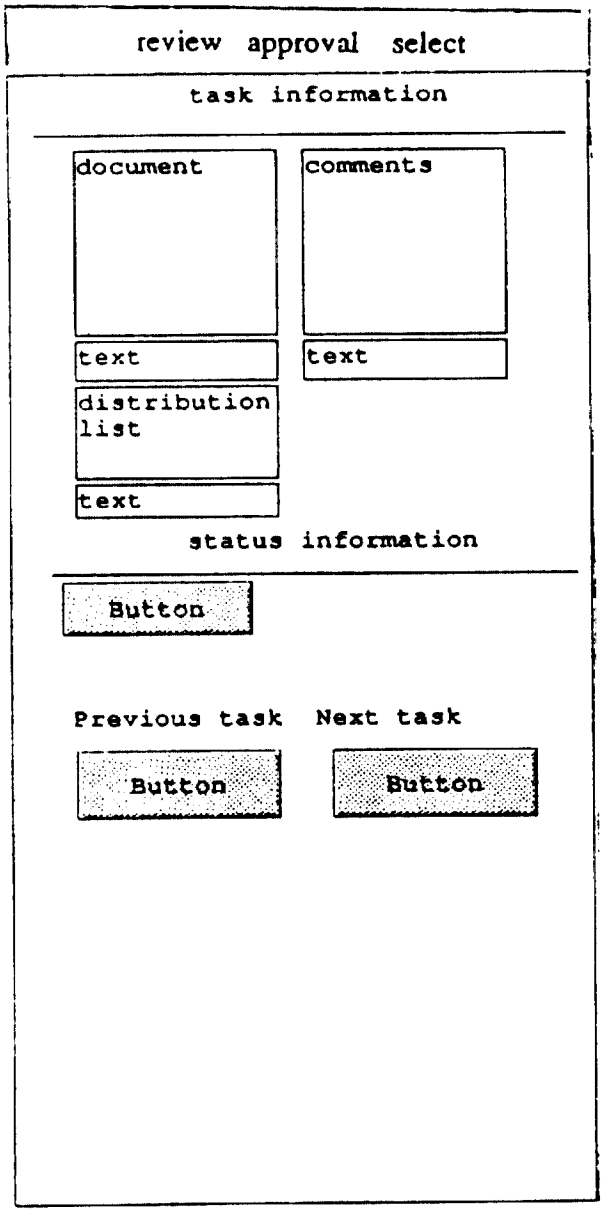

Figure 13: Sketch of Display Generated for "approval"

```
The process being described is:el-rv-ap
this process if from the viewpoint of:NASA
task1 in this process is:review
Task2 is :approval
Task3 is :select
the current task is:review
this information is needed:documents
the information is to be presented as:text
the importance of this information is primary
the current task is:review
this information is needed:comments
the information is to be presented as:text
the importance of this information is primary
the current task is:review
this information is needed:distribution list
the information is to be presented as:text
the importance of this information is secondary
the current task is:review
this information is needed:list of files
the information is to be presented as:text
the importance of this information is secondary
the current task is:review
this information is needed:status of change
the information is to be presented as:text
the importance of this information is status only
the following actions are performed on this info:redline
the following actions are performed on this info:comment
the following actions are performed on this info:distribute
the following actions are performed on this info:display
the following actions are performed on this info:save
the following actions are performed on this info:compare
the following actions are performed on this info:print
the previous task is:select
the task that follows this one is:approval
the current task is:approval
this information is needed:document
the information is to be presented as:text
the importance of this information is primary
the current task is:approval
this information is needed:comments
the information is to be presented as:text
the importance of this information is primary
the current task is:approval
this information is needed:distribution list
the information is to be presented as:text
the importance of this information is secondary
```

Figure 14: Portion of Data Generated for Review and Approval Process

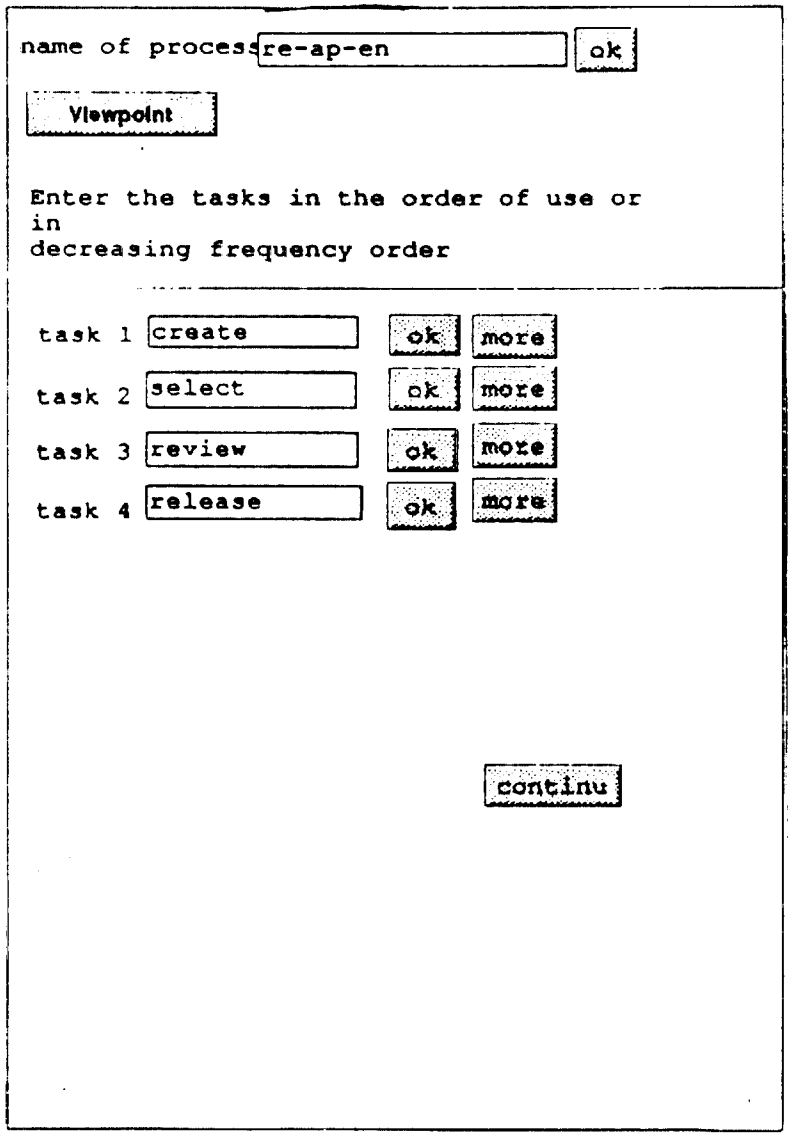

Figure 15: Viewpoint 2: Review and Approval from Engineering View

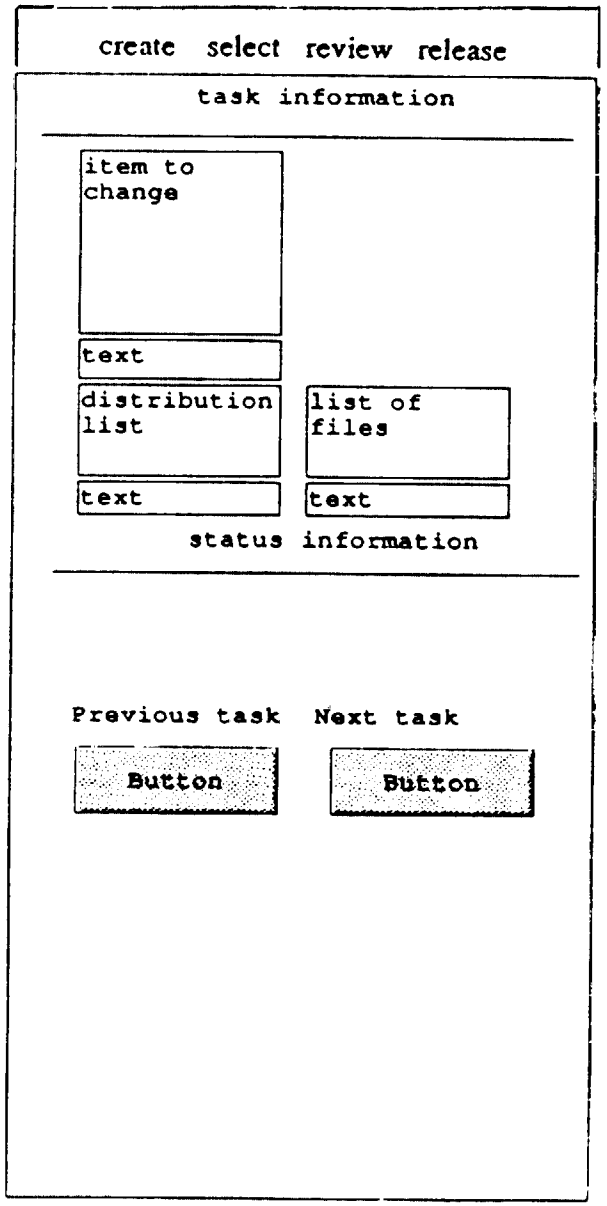

Figure 16: Viewpoint 2: Sketch of Interface for "create"

V. Future Plans for Testing and Using TAT

5.1 Uses for TAT Output

As was previously stated, the TAT output is meant to serve several purposes. First of all, the sketched interface serves to give a more concrete aspect to the task analysis in a form that is easily understood by the user. Using this sketch, the user should be able to assess it for completeness and correctness. The interface could be used in a representative scenario of the process which the user could work through. This sketch should accompany functional requirements given to the developers to facilitate design of the user interface.

Analysis programs could be written to scan the data files generated. This would be particularly useful in the case where several viewpoints are being examined or where several applications are to be run concurrently. The data files can be examined to see conflicts and commonalties in information sources and presentation methods. In particular, common tasks or similar tasks should possess similar actions. Consistency in interface design has been recognized as beneficial to success of software companies (Tognazzini, 1989). Consistency in presentation and actions can be analyzed using the TAT data files.

5.2 Additions to TAT

The prototyped version of TAT is a very rough version. There is much work yet to be done on determining what kinds of information should be collected. Information about feedback desired from a given action seems a likely candidate as does information on the frequency and duration of the task. In order to determine the completeness of information collecting in TAT it will be necessary to try it out in many different domains.

5.3 Functionality Needed

There are many functions that need to be included in a coded version of TAT. The functionality of the current version is limited due to the nature of the prototyping tool used to implement it and the time limitations during which TAT was constructed. Functionality that is seen as needed includes:

1. The ability to display labels on status buttons and task link buttons in the interface sketch.
2. The ability to link up tasks in multiple paths.
3. The ability to save and display sub menu items in the interface sketch. Currently this information is saved in the data file but once the interface sketch is closed, they do not appear in the saved sketch.
4. There should be a way to associate actions with a particular piece of information. This type of knowledge could be useful if deciding to break the task into several displays in the final design of the interface.
5. The user should be able to easily change the choices displayed in the dialog boxes on viewpoint and information type. These choices are

dependent on the domain in which TAT is being used. In addition the user should be able to easily change the parameters concerning the number of tasks and information sources.

5.4 An Initial Test of TAT

The information used in generating this example was produced mainly from informal interviews with personnel involved in Shuttle Flow Processing. This was due mainly to the limited time frame for development of the prototype. However, a paper version was used in one instance to obtain information about the review and approval process. Several observations were made during this process. First, a new step in the review process, that of comparing initial changes and comments to the newly distributed change, was identified. Perhaps this step would have eventually been discovered through further interviews but having to simplify one's thoughts about the task and flow seemed to clarify the process.

The ability to be able to distribute the change to a person other than the originator was identified as was the capability of seeing which jobs were currently being worked when reviewing changes.. While TAT does not currently capture all this information it is rewarding that using this approach elicited this information. This suggests that using TAT along with note taking or audio/video recordings would be a beneficial approach.

5.5 Testing

In order to determine how useful TAT is, it must be used in the development of several prototypes and these compared to the prototypes developed without this tool. In addition, it needs to be determined what kinds of analysis should be performed on the data files and what, if any, other information should be collected that will be useful. It is expected that TAT will evolve as it is used in more varied domains. Testing the benefits of using TAT will be a difficult task. In the best scenario software would be developed with and without using TAT. Performing these kinds of parallel developmental tests in the real work are difficult if not impossible. Therefore, the most realistic situation would be to use it in as many varied situations as possible and use feedback from the users, developers and HCI personnel to determine the benefits.

VI. Conclusions

Development of good interfaces in software means the ability to closely map the user's task to interface elements. This depends on producing a good task analysis and upon an iterative design process. Unfortunately there are obstacles to being able to accomplish both of these. Producing a good task analysis is especially difficult in cases where the domain is complex and in which much user training is needed. The person conducting the task analysis is often given information from the user with no way of assessing its completeness or its relative importance. Moreover, being able to translate this information into an initial prototype is difficult. This is especially the case in situations where no system is currently in place.

In addition it is important to have the ability to communicate to the developer the user's expectations of an interface as early as possible in the design cycle. This helps to reduce the iterative design process and hence lessen efforts and costs.

The Task Analysis Tool is a step in the proper direction. Although simplistic in nature, it serves to obtain feedback from the end user at an early point in the design cycle. This feedback can be easily communicated to software designers to use as a basis for initial prototypes and interface designs. Further refinements of the Task Analysis Tool will be done. Then its benefits in facilitating interface development will be assessed.

VII. References

1. Asymetrix Corporation. (1989). Toolbook. Computer Software.

2. Boker, S. and Gronbaek, K. (1991). Cooperative prototyping: users and designers in mutual activity. *International Journal of Man-Machine Studies*, 34, 453-478.

3. Drury, C., Paramore, B, Van Cott, H., Grey, S and Corlett, E. (1987). Task Analysis. In Salvendy, G. (Ed.), *Handbook of Human Factor*. (pp. 370-401). New York: Wiley and Sons.

4. Grudin, J. (1991). Obstacles to user involvement in software product development, with implications for CSCW. *International Journal of Man-Machine Studies*. 34, 435-452.

5. Guindon, R. (1988). *Cognitive Science and It's Application for Human-Computer Interaction*. Hillsdale, NJ. Lawrence Erlbaum Associates.

6. Maddix, F. (1990). *Human-Computer Interaction: Theory and Practice*. West Sussex, UK: Ellis Horwood Limited.

7. Norman, D. (1986) Cognitive engineering. In D. A. Norman & S. W. Draper (Eds.), *User centered system design*. (pp. 31-61). Hillsdale, NJ: Lawrence Erlbaum Associates.

8. Shackel, B. (1988) . Introduction to the design of end-user interfaces. In N. Heaton and M. Sinclair (Eds.) *Designing end-user interfaces*.(pp. 97-109). Maidenhead Bershire, UK: Pergamon Infotech Limited.

9. Tognazzini, B. (1989). Achieving Consistency for the Macintosh. In J. Nielsen (Ed.) *Coordinating User Interfaces for Consistency*. (pp. 57-74). San Diego, CA: Academic Press.

10. Walsh, P., Lim, K. , Long J. and Carver, M. (1988). Integrating human factors with system development. In N. Heaton and M. Sinclair (Eds.) *Designing end-user interfaces*.(pp. 111-119). Maidenhead Bershire, UK: Pergamon Infotech Limited.

11. Wilson, J. and Rosenberg, D. (1988). Rapid Prototyping for User Interface Design. In M. Helander (Ed.) *Handbook of Human-Computer Interaction*. (pp. 859-875). Amsterdam: Elsevier Science Publishers (North Holland).