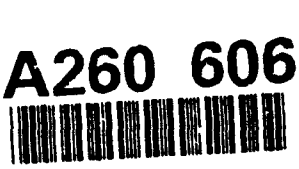

(2)

Measures of User-System Interface Effectiveness: An Encoding Scheme and Indicators for Assessing the Usability of Graphical, Direct-Manipulation Style User Interfaces

MTR 92B0000047V3
January 1993

Donna L. Cuomo
Charles D. Bowen

MITRE
Bedford, Massachusetts

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	January 1993	

4. TITLE AND SUBTITLE
Measures of User-System Interface Effectiveness: An Encoding Scheme and Indicators for Assessing the Usability of Graphical, Direct-Manipulation Style User Interfaces

5. FUNDING NUMBERS

6. AUTHOR(S)
Donna L. Cuomo
Charles D. Bowen

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
The MITRE Corporation
202 Burlington Road
Bedford, MA 01730-1420

8. PERFORMING ORGANIZATION REPORT NUMBER
MTR 92B0000047V3

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
same as above

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
same as above

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

See attached.

14. SUBJECT TERMS
User-system Interface, Human-computer Interaction

15. NUMBER OF PAGES
92

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	Unlimited

Measures of User-System Interface Effectiveness: An Encoding Scheme and Indicators for Assessing the Usability of Graphical, Direct-Manipulation Style User Interfaces

MTR 92B0000047V3
January 1993

Donna L. Cuomo
Charles D. Bowen

Contract Sponsor MSR
Contract No. N/A
Project No. 9162A
Dept. D047

MITRE
Bedford, Massachusetts

Department Approval: Nancy C. Goodwin.
Nancy C. Goodwin

MITRE Project Approval: Donna L. Cuomo
Donna L. Cuomo

ABSTRACT

The purpose of this MITRE Sponsored Research project was to develop methods and measures for evaluating user-system interface effectiveness for command and control systems with graphical, direct manipulation style interfaces. Due to the increased use of user interface prototyping during concept definition and demonstration/validation phases, the opportunity exists for human factors engineers to apply evaluation methodologies early enough in the life cycle to make an impact on system design. Understanding and improving user-system interface (USI) evaluation techniques is critical to this process. In 1986, Norman proposed a descriptive "stages of user activity" model of human-computer interaction (HCI). Hutchins, Hollan, and Norman (1986) proposed concepts of measures based on the model which would assess the directness of the engagements between the user and the interface at each stage of the model. We created operational definitions of the concepts of directness, and derived observable indicators that certain types of indirectness may exist in the interface design. This phase of our research program involved using these concepts as a basis for a methodology of analyzing data collected during usability studies. A usability study was performed on the Military Airspace Management System (MAMS) prototype; four participants' and one user interface expert's data were used for further analysis.

We first proved that in order to assess concepts such as the directness of user-system interface engagements we need to know both what the user intended to do and what they did. This involves integrating data collected via different media (computer collected keystrokes, transcribed user protocols, video of the display output). A model-based, two-level encoding scheme was then created and applied to the usability data to aid in extracting and quantifying measures of USI effectiveness. The first level provides a high-level description of user activity, depicting users' task intentions, intentions to execute, errors by stages, and the success of their endeavors. The second level provides detailed information on the users' input activities at a user-interface object level. The two levels combined provide a complete description of what the users want to do, how they did it, and how directly the system allows them to do it. We then manually extracted our derived indicators of indirectness from each user's data and were able to perform a much more complete and quantifiable analysis of the user-system interface than would have been possible with more traditional evaluation methods. Examples of usability problems identified with this method are provided and we discuss the need for a computer tool to make application of the method more efficient.

EXECUTIVE SUMMARY

INTRODUCTION

The focus of the project Measures of User-System Interface Effectiveness is to study and validate methodologies and measures for analyzing the overall effectiveness of user-system interfaces (USI) for task performance. There is an increased emphasis on user-centered system design which involves designing a system from a user's perspective, where the concepts, objects, and actions embodied in a system closely match the user's task concepts, objects, and actions allowing users to interact with the computer task domain in a direct way. This report, the third in a series of MSR reports, documents the evaluation methodology we developed for analyzing data collected in usability studies, and provides examples of the method applied to a prototyped system.

MEASURING GRAPHICAL, DIRECT MANIPULATION STYLE INTERFACES

The class of interfaces we were interested in evaluating were graphical, direct-manipulation style interfaces supporting ill-defined tasks. Ill-defined tasks refer to tasks which users perform which have more than one correct solution, and alternative methods for performing these tasks exist. This class of applications would include scheduling tasks, mission planning tasks, and computer-aided architectural design tasks. These tasks can be contrasted to well-defined tasks such as some data entry tasks where there is one correct solution, e.g., a document is entered into the system and edited until error free. The attributes of the interface, direct manipulation and graphical, as well as the ill-defined nature of the tasks makes traditional USI evaluation measures less useful in terms of the feedback they provide. Traditional USI evaluation measures tend to be summary measures such as time to complete a task, percent of task completed, time spent in errors, percent or number of errors, command frequency, etc. (Whiteside et al., 1988). These are gross measures and while various aspects of the interface will undoubtedly affect these measures, this type of measure alone does not provide us with enough granularity and diagnostic information on each user interaction with the system. Additionally, the concepts of direct manipulation raise a virtually unexplored area in terms of defining and measuring directness to a degree that they can be applied in practice. In summary, a method for assessing user interfaces for this class of interfaces needs to be defined.

CONCEPTS OF SEMANTIC AND ARTICULATORY DISTANCE

Norman (1986), and Hutchins, Hollan and Norman (1986) provide a good treatment of concepts of directness in user-system engagements. In their conceptual model of human-computer interaction they describe seven stages a user could traverse while accomplishing a goal with a computer: intention formation, action specification, execution, perception, interpretation, and evaluation. They then define four concepts of distance which are critical to making a design user-centered: semantic and articulatory distance of execution, and semantic and articulatory distance of evaluation. Semantic distance of execution spans the intention formation stage and involves whether the user can say what he/she wants to say directly with the computer system or whether a complex expression is required. Articulatory distance of execution spans the action specification stage and reflects the closeness of the form of the action to be executed to the meaning of the input expression. This is followed by the stages of execution and perception -- the stages spanning the translation from mental state to physical activity and back again. Articulatory distance of evaluation spans the interpretation stage and concerns how easily the output expression can be extracted from the output expression form. Semantic distance of evaluation concerns the ease with which users can determine whether they accomplished their goal.

These concepts are complex and intriguing but still rather high-level. Characterizing a system by how well it supported the different stages, however, would provide us with the right level of information needed to successfully iterate a design. We derived indicators or behaviors of indirectness for each stage, based on Hutchins et al. concepts of directness; one set of indicators is shown below. Supporting identification of the indicators involves collecting and evaluating user-system performance at an interaction-by-interaction level and the sequencing of engagements would be important. We derived a model-based methodology which allows us to do this.

Causes of semantic indirectness of execution and evaluation and the corresponding observable indicators

Semantic indirectness of execution if:	***Indicator***
User intention not supported	• Protocol stating desired function • Attempting to execute unsupported function, forced to abort
Missing high-level object	• Same step or set of actions repeated on lower-level objects
Complex expression required to accomplish intention	• Many steps/actions required to complete intention • Errors in step order • Incomplete/aborts in intentions

Semantic indirectness of evaluation if:	*Indicator*
Extra step/s required to perform an evaluation	• Number and purpose of steps performed (e.g., to get information, or "check" something)
Difficult or user unable to perform an evaluation	• Frequency and types of evaluation errors • Evaluation not made

THE METHODOLOGY

The methodology consisted of four major steps. The first step was to conduct a usability study; this involves real users exercising a system or prototype while evaluators collect data on the process. We have determined that both verbal protocol data (where users are asked to voice their thoughts aloud), as well as time-stamped computer collected history logs (records all the users input actions) are required to be able to assess the four directness of engagement concepts. Protocols provide information about what a user intends to do while the history log provides information about how the user did it. The latter is easier to collect and analyze but is ambiguous and insufficient if used alone.

A usability study was conducted using a prototyped airspace management scheduling system. Data was collected on seven participants, with the method being applied to five of the participant's data. One of the participants was the USI design engineer for the project and served as our "user-interface expert" participant.

The second step involved integrating the collected data by combining the transcribed user protocols with the appropriate portions of the user's history file; this was done manually.

The third step involved developing and applying a two-level encoding scheme, based on Norman's model, to the data. The first level of the encoding scheme provides a high-level description of user activity, depicting users' task intentions, intentions to execute, errors by stages, and the success of each endeavor. The second level provides detailed information on the users' input activities at a user-interface object level. The two levels combined provide a complete description of what the users want to do, how they did it, and how well they did it. The codes and their descriptions are shown in the tables below.

Semantic-Level Encodings

Encoding	*Definition*
Goal	Scenario step.
Task intention (Int.task)	An intention to complete one task contributing to the completion of a goal.
Perception intention (Int.per)	An intention to improve the perceptibility of a display.
Intention to execute (Int.exe)	One computer step (may be comprised of multiple actions) leading to the completion of a task intention. Several steps may be required per task intention.
Evaluate (Eval)	The success with which the intention was accomplished.
Error in intention (Err.int)	The intention was incorrect and will not accomplish the goal.
Error in action specification (Err.acsp)	Wrong sequence of actions to accomplish the intention to execute.
Error in execution (Err. exec)	Manual, motor error in executing.
Error in perception (Err. per)	Break-down in human perceptual processing of information on a display.
Error in interpretation (Err. inter)	User fails to interpret system state correctly.
Error in evaluation (Err.eval)	User mistakenly thinks has or has not moved closer to the goal.
Recovered error (Rec.err)	Error was detected and recovered from.

Articulatory-Level Encodings

Encoding	*Definition*
Menu	A menu was opened
Command	A command was selected
List-Select	An item was selected from a list
Button	A button was selected
Field	An action was taken in a field
Scroll	A scroll bar action was performed
Window	A window action was performed
Application-specific objects	Encodings to track the manipulation of application-specific objects

The encoding of the data was done with the aid of a tool called SHAPA, developed at the University of Illinois at Urbana-Champagne.

The fourth step in the evaluation methodology involved extracting the indicators of interest from the encoded data files and comparing them across users. For ease of recording the extracted information, we created a data summarization table. For each user task intention the critical information is summarized in a manner which allows for easy comparison across subjects.

An excerpt from a real participant's summary table for the task "schedule missions" is shown in the figure below.

Int.task	Freq	Int.exec	Freq	# actions per int.exec	Eval of int.exec	Eval of int.task	Errors	Com-ments
2-24 resconf thawk/sdt-w	1	lookthawk/sdt-w lookwpn-w movesdt-w	1 1 1	4 1 6	OK OK OK	 OK		
2-25 schsdt-w	1	schsdt-w	1	3	OK	OK		
2-26 schwpn-w	1	schwpn-w	1	2	OK	OK		
2-27 sch1240026-W	1	look1240026-w sch1240026-w	1 1	2 1	OK OK	 OK		
2-30 schfox-w	1	lookfox-w movefox-w schfox-w	1 1 1	2 1 1	OK OK OK	 OK	e14 - conflict state - R err.inter	

Additionally, we collected the same data and completed the same form on the system user-interface expert, to provide us with a baseline of expert system performance. The expert's data illustrates the best the system can do. The real users' data illustrates the ease with which the users could accomplish their tasks with this system and the directness of engagements. Comparison of data across subjects allows for distinguishing between system-induced problems (more than one user has same difficulty), effects of training (only least-experienced subjects had the problem), and individual user problems (only one user had that type of problem). Examples of the type of information we were able to extract for one goal, "schedule missions", are summarized below.

Indicator	*Potential Problem*
Repetitive sequences for applying the approve command to missions	Can not select groups of objects for application of a single command
Repetitive sequences for applying the approve command to parts of a single mission	System does not consider mission as an object for the case of applying scheduling commands
An abort while trying to bring up all parts of the "dact" mission on the display	System does not consider mission parts as an object for the case of finding the whole mission
An extra intention to execute required to "look" when the task intention is to schedule a mission	Information in dialog box is often required before mission can be approved. To increase feeling of directness the two steps should be combined in some manner.
Perceptual/execute errors	When the missions were too close together, users would select the wrong one. There was no way to differentiate missions when the labels were very small.
Execute error, many actions for recovery	A user selects deny from menu rather than approve which is adjacent. Lack of undo causes user to perform multiple actions to fix.
	Mission icons were often accidentally moved. Users then had to manually reposition them. Two problems are icons were too sensitive, and there is a lack of an undo feature, resulting in multiple actions to undo a previous action.

Finally, an error taxonomy based on the stages in the HCI model was created and applied to the recorded instances of errors. This information is provided in Appendix D.

ACCOMPLISHMENTS

The results obtained to date on measures of user-system interface effectiveness are very promising. We developed a method which allows us to obtain information on the directness of user engagements with a system. The method involves integrating protocol data with history file data, for a complete and useful picture of human-computer interaction (HCI) activity. We created a theory-based encoding scheme which provides a method for quantitative analysis of the data. We also created an error classification scheme based on the stages of user activity model, which provides much more information on why an error occurred and how to fix the system to prevent it than is possible to obtain from traditional error frequency measures. Indicators of user-system interface (USI) effectiveness extending beyond errors and time were derived and found to be useful. We have shown that a USI engagement can be error-free but not be direct, and new measures and indicators such as those described here are required for a complete evaluation. The measures are also in a form which allows for easy comparison across subjects. This technique also allows us to determine whether difficulties are due to a single user's inexperience or whether problems can be attributed to the system design.

FUTURE WORK

Measures and Indicators

We need to do several things in the area of refining the USI measures and indicators. First, we need more rigorous definitions of the different levels of the encodings; when, for example, is something a task intention as compared to an intention to execute? While we tried to be consistent in our application of these terms, it was difficult, particularly as this was the first time we applied the scheme. The same problem holds with regard to the level of detail for the intentions to execute. Sometimes all actions within a dialog box were considered to be a single intention to execute, and sometimes particular actions were broken out separately. This may need to be flexible based on what areas of the USI are to be evaluated.

We would like to continue to work on the definitions and names for the different USI indicators. There also seem to be many kinds of repetitions which are indicators of different types and levels of problems. We would like to classify all of these various kinds of repetitions and determine what they imply for the system design. Finally, we need to apply the encoding scheme to a different system to ensure it is generic across systems, and continue to refine it.

The time required to apply our evaluation method could be greatly shortened by creating a software system dedicated to supporting this method. The system will be multi-media and will aid in integrating the history file and the users' intentions. We may be able to remove the step of transcribing all of the users' protocols and just extract the information needed for the intentions, evaluations, etc. We currently plan to build this tool in FY93.

ACKNOWLEDGMENTS

The authors would like to acknowledge Scott E. Blomquist for providing the software engineering support and expertise for this project. Scott created a robust version of the Military Airspace Management System prototype which was the software used for evaluation during our study. Scott also incorporated data collection routines into the software, allowing time-stamped history logs of all user actions to be recorded. He later created filtering software to extract the data of interest from that collected. We would also like to acknowledge the hard work and efforts of Elizabeth Wadick, the co-op student who supported this project. Elizabeth did many activities in support of this project but her biggest contribution was in her data analysis efforts. Elizabeth manually transcribed all the subjects' comments from the video tapes, she integrated the various data files, and she painstakingly applied the encodings to the data and entered the information into our software analysis package. Finally, we would like to thank the reviewers of this paper, Nancy C. Goodwin and Gabriel Spitz, for their helpful comments which improved the quality of the report.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

SECTION 1

INTRODUCTION

The focus of the project Measures of User-System Interface Effectiveness is to study and validate methodologies and measures for analyzing the overall effectiveness of graphical direct-manipulation user-system interfaces (USI) for task performance. There is an increased emphasis on user-centered system design. This involves designing a system from a user's perspective so that the concepts, objects, and actions embodied in a system closely match the user's task concepts, objects, and actions, thus allowing users to interact with the computer task domain in a more direct way. There has been little work done on operationally defining concepts of directness or defining evaluation techniques for assessing directness. This paper documents the results of using an existing human-computer interaction framework to develop cognitive-based measures of the directness of engagements, and the results of applying this new approach to an actual system. Section 1 provides background information on the project, the rationale for this work, and introduces the theories and models on which the work was based. Section 2 discusses the new scheme for assessing graphical user interfaces at a prescriptive and cognitive level. Section 3 provides the results of applying the new evaluation methodology to a prototyped system. Section 4 concludes with a review of the advantages of the new method and discusses additional areas of research required, particularly the need for a tool to support application of the method.

1.1 FY92 RESEARCH PROGRAM

The plan for the FY92 MSR project is shown in figure 1. The first step was to review models of human-computer interaction (HCI), review user-system interface (USI) evaluation techniques and data analysis tools, and to derive concepts for potential USI measures based on the models of human cognition and HCI. Volume I of this MSR report series (MTR 92B0000047) documented the results of this first phase of the research. From the review of USI models, measures, and evaluation techniques, we decided to focus further study on structured judgement techniques and user-based evaluations as these appeared most suitable for application to command and control (C2) systems. The issue to be resolved with the structured judgement techniques was whether they assessed USI problem areas on the newer graphical, direct-manipulation style interfaces. For user-based evaluations, the concern was whether traditional measures captured USI effectiveness of the graphical, direct-manipulation-style interfaces. The second and third phases of the research project addressed these issues.

In phase 2, structured judgement evaluation techniques were reviewed and applied to predict where users might encounter interaction difficulties during task performance. Structured judgement techniques involve an expert in USI design assessing a user interface; users are not involved. These evaluation techniques were applied to the Military Airspace Management System (MAMS), a prototype of a military airspace scheduling system. This system served as our application system for the entire study and was selected because it has a

graphical, direct-manipulation style interface. The three structured judgement techniques used were cognitive walkthrough, heuristic evaluation, and guidelines. It was found that the cognitive walkthrough method applied almost exclusively to analyzing the user's computer-input actions. The guidelines were more generally applicable across the various stages of human-computer interaction but all the techniques were weak in measuring activities involving display perception, interpretation, and evaluation, and how similar the concepts encoded by the computer language are to the way users think about these concepts. The conclusion was that improvements to existing or new techniques are required for evaluating the directness of engagements for graphical, direct manipulation style interfaces. Phase 2 of this research was documented in Volume II of this MSR report series (MTR 92B0000047V2).

Finally, the third phase of the research involved conducting a user-based evaluation and developing new measures of USI effectiveness for analyzing data collected during such evaluations. These results are documented in this report. Below we discuss why measures and indicators of USI effectiveness are needed, and we introduce the concept of direct manipulation style interfaces. This is followed by a review of Norman's and Hutchins, Hollan and Norman's theories on stages of user activity and the concepts of semantic and articulatory distance.

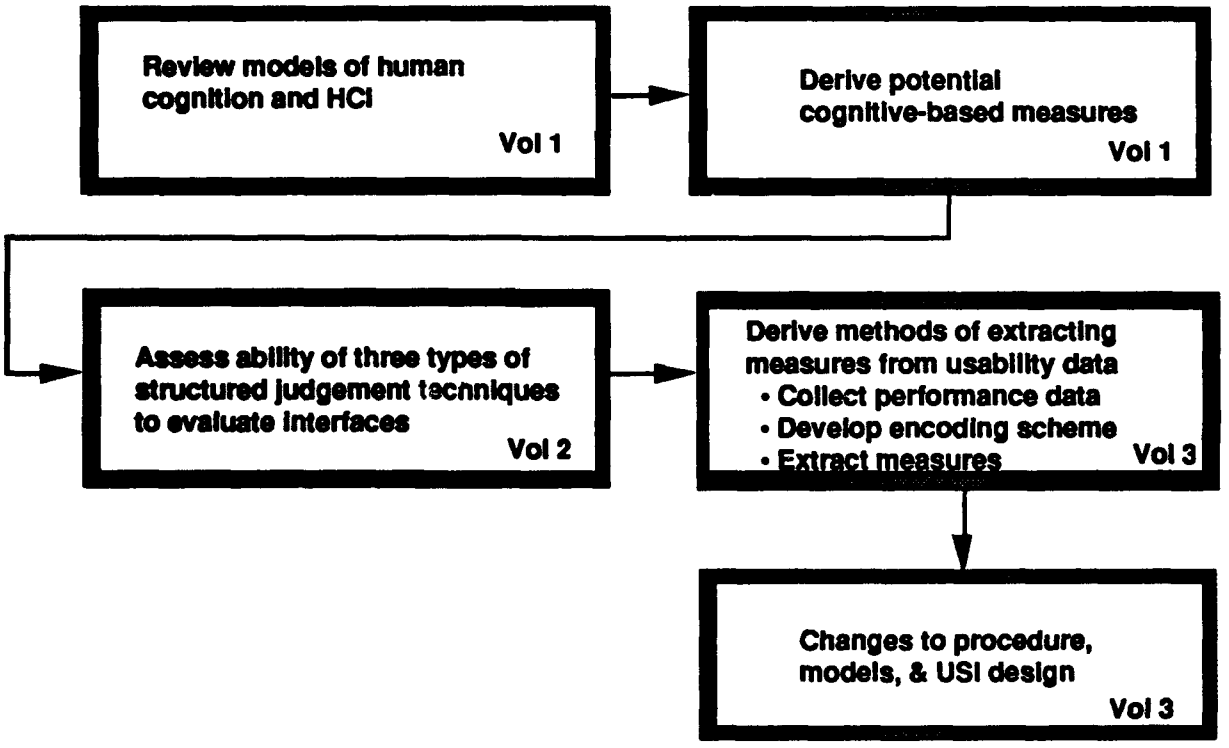

Figure 1. Overall Research Plan.

1.2 RATIONALE FOR COGNITIVE-BASED USI EFFECTIVENESS MEASURES

Several changes in the area of computers and user-system interface design bring about the need for cognitive-based measures of USI effectiveness. Computers are increasingly used to support dynamic, interactive tasks in which the user's mind is an important component of the total system (Landauer, 1987). No longer are users using a computer as a tool to perform independent tasks, by giving instructions to the computer and waiting for a reply. Object-oriented programming allows the task domain world to be graphically presented by the computer for direct interaction with the user. As more and more applications adopt graphical interfaces we need a cognitive-based method of evaluating such user interfaces to measure their effectiveness.

A primary usability feature of a graphical user interface is the directness with which a user can manipulate and control a software system. Schneiderman (1982, 1983, in Ziegler et al., 1988) characterizes directness as:

- Continuous representation of the object of interest
- Physical actions or labeled button presses instead of complex syntax and command names
- *Rapid incremental reversible operations* whose impacts on the object of interest are immediately visible.

These types of interfaces are hypothesized to be "easier to use" than dialogue-style interfaces because the basic functionality can be quickly learned and actions are immediately reversible. This does not, however, explain how directness of a user interface be designed and evaluated.

The concept of direct manipulation (DM) is relatively complex (Hutchins, Hollan, and Norman, 1986). Hutchins et al. note that despite the promise of this concept, there is no account of how particular properties might produce the feeling of "directness." Intuitively, directness can be described in terms of the type and number of mental steps required by the user to achieve a desired goal. For example, if we want something to be displaced by two inches, we move it two inches; our actions directly mimic our intentions.

The high-level issues relating to directness and evaluating directness are:

- Does the computer represent the task domain in the same way the user conceptualizes the task domain?
- Are the objects in the system at the level of task object the user expects?

- Is the USI directly supporting the user's cognitive processes during task performance?
- Is the user forced to relearn how to do a task to suit the computer's model of the task domain?
- Does the USI use terminology and icons unfamiliar to the user?
- Can poor/good interactive sequences between the human and the computer be identified?

Norman (1986) provides a generic model or framework which describes the mental steps associated with the execution of a higher level goal. This framework and its related concepts, presented below, provide a means of discussing and structuring the issues associated with directness.

1.3 STAGES OF USER ACTIVITY MODEL

When examining human-computer task execution, Norman (1986) discusses the discrepancy between a human's psychologically expressed goals and the physical controls and variables of a computer system. Once a human has a goal (a state a person wishes to achieve), the human must translate this goal into the desired system state, determine what settings will achieve this state, and then determine what physical manipulations are required. Following the execution of the required manipulation and system response, the human must then transform the physical variables of the system state into expressions which are relevant to the psychological variable of the goals. This is a feedback loop where the results of one activity are used to direct further activities.

This theory underlies Norman's stages of user activity model. Norman took the approach of dividing up user tasks into distinct segments. He uses the metaphor of bridging a gulf to describe the issue of dealing with the psychological and physical variables in computer system design and evaluation. The "Gulf of Execution" represents the gap between psychological goals and the physical system. The four segments which bridge the "Gulf of Execution" are: intention formation, specification of an action sequence, executing an action, and making contact with the data entry mechanisms of the user interface (figure 2).

The "Gulf of Evaluation" represents the gap between the physical system and the original psychological goals and intentions. The four segments which bridge the "Gulf of Evaluation" are: user interface display output, perceptual processing of display output, interpretation of display output, and evaluation or the comparison of the system state as represented by display output to the original goals and expectations (figure 2). Note that "display output" could be auditory or even tactile; it might not be limited to visual display.

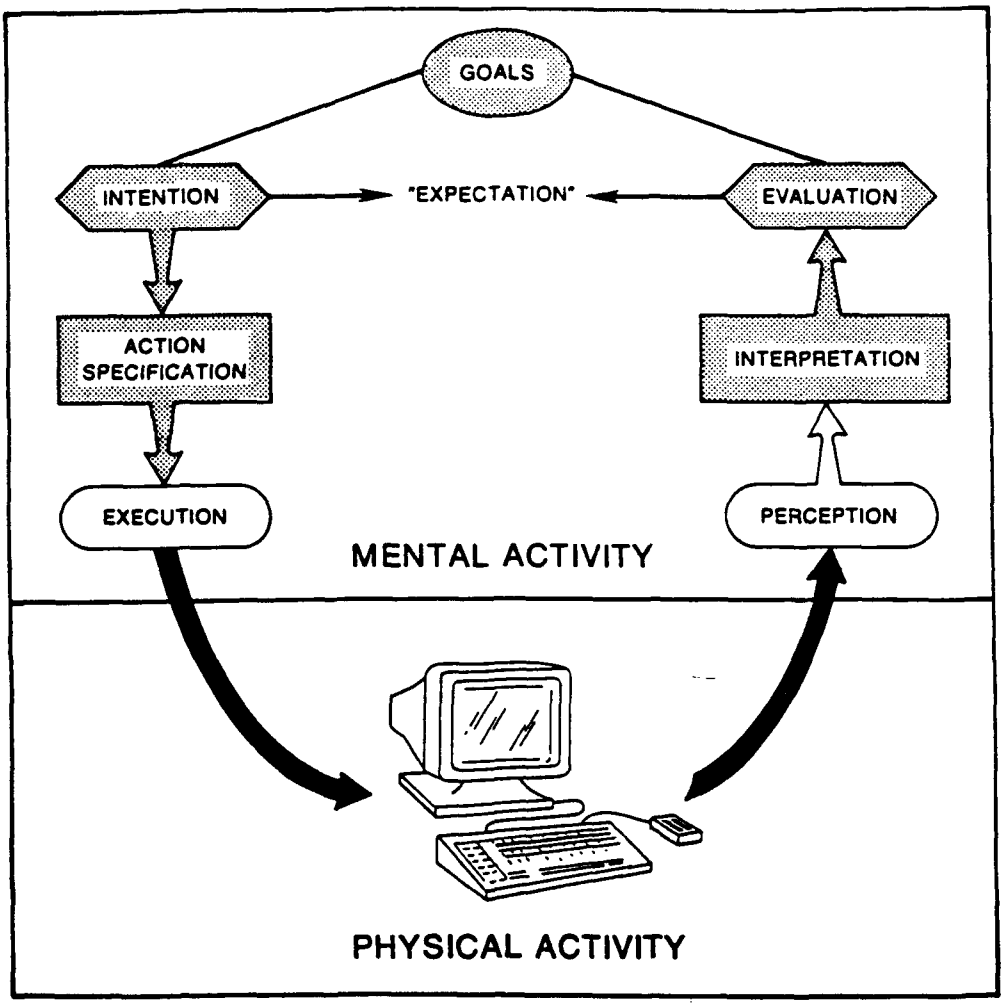

Figure 2. The Stages of User Activity

From *User Centered System Design: New Perspectives on Human-Computer Interaction* (p.42) by D. A. Norman and S. W. Draper, 1986, Hillsdale, New Jersey: Lawrence Erlbaum Associates, Inc. Copyright 1986 by Lawrence Erlbaum Associates. Reprinted by permission.

Forming an intention is the activity that specifies the meaning of the input expression that is to satisfy the user's goal. The action specification prescribes the form of an input expression having the desired meaning. These two activities are psychological activities. The form of the input expression is then executed by the user on the computer interface and the form of the output expression appears on the display, to be perceived by the user. Interpretation determines the meaning of the output expression from the form of the output expression. Evaluation assesses the relationship between the meaning of the output expression and the user's goals (Hutchins, Hollan, and Norman, 1986). The last two stages are also psychological activities.

Norman concedes that real activity does not progress as a simple sequence of stages. Stages of activity sometimes appear out of order and some stages are skipped while other stages may be repeated. The stages on the evaluation side, for instance, can be occurring almost continuously at some level during an interaction sequence.

1.4 CONCEPTS OF DISTANCE AND DIRECTNESS

Hutchins et al. (1986) further elaborated Norman's model and suggested that directness can be derived from the degree of mental transformation required to span one's thoughts about the task and the physical requirements of the system, and the qualitative feeling that one is *directly* manipulating the objects of interest in the task domain. They termed these concepts semantic and articulatory distances of execution and evaluation. Below we describe these concepts, which are also illustrated in figure 3.

1.4.1 Semantic Distance/Directness

Semantic distance of execution involves matching the level of description required by the interface language to the level at which the person thinks of the task. Distance involves how well the computer language encodes concepts as the user thinks of them: can a concept be expressed directly or is a complicated expression required? The resulting semantic distance can be measured by how much of the required structure is provided by the system and how much by the user; the more the user provides, the greater the distance (Hutchins et al., 1986). As distance increases, directness lessens. Semantic distance is related to the "nouns" and "verbs" or "objects" and "actions" provided by a computer system. Suppose, for example, a user of a scheduling system wishes to schedule Crew A to fly a mission. If the system only provides individual crew members as the objects the user can manipulate, the user would have to identify the individual members of crew A and repeatedly perform the action which schedules each crew member for a mission, resulting in a complicated expression to achieve a simple concept. The user is being forced to work at a lower level than desired, resulting in greater semantic distance. For execution, forming an intention is the activity that spans semantic distance. The intention specifies the meaning of the input expression that is to satisfy the user's goal or subgoal.

Semantic distance also occurs on the evaluation side of the interaction cycle. Here, semantic distance is proportional to the amount of processing required by the user to determine whether the goal has been achieved. If the terms of the computer output do not match those of the user's intention or output is lacking, translation or inference by the user is needed, increasing the semantic distance (Hutchins et al., 1986). Using the same example as above, if the user was allowed to schedule "crew A", but the output of the action showed only individual crew members who were scheduled, and not grouped or identified by crew name, the user might be uncertain as to whether his/her intention was fulfilled. For evaluation, the stage of evaluation spans semantic distance.

Instances where a system does not support a user intention or evaluation, even indirectly, are also of interest.

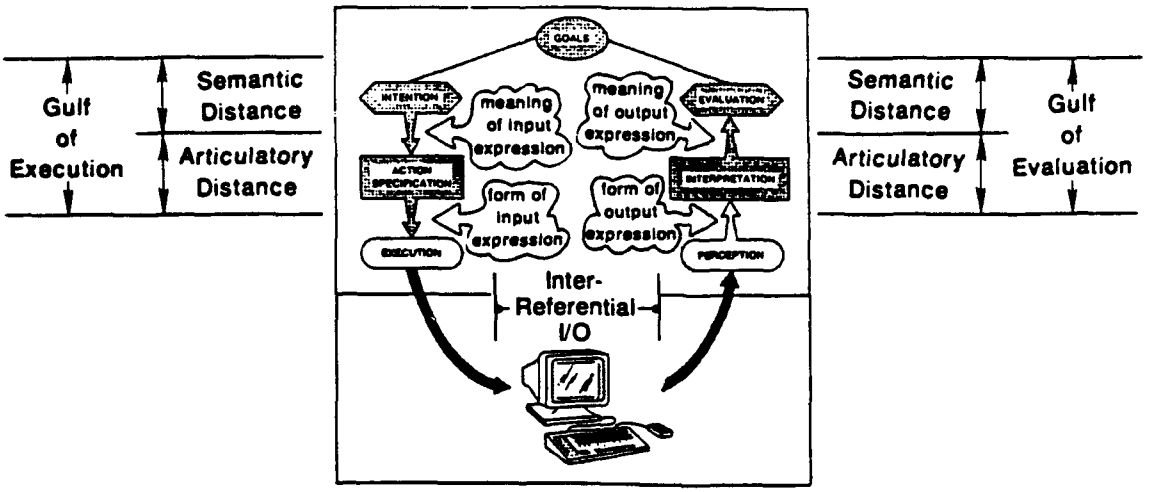

Figure 3. Semantic and Articulatory Distance of Execution and Evaluation
From *User Centered System Design: New Perspectives on Human-Computer Interaction* (p.111) by D. A. Norman and S. W. Draper, 1986, Hillsdale, New Jersey: Lawrence Erlbaum Associates, Inc. Copyright 1986 by Lawrence Erlbaum Associates. Reprinted by permission.

1.4.2 Articulatory Distance/Directness

Whereas semantic distance relates to relationships between users' intentions and meanings of expressions, articulatory distance relates to relationships between the meanings of expressions and their physical form (Hutchins et al., 1986).

On the execution side, the form may be keystrokes, mouse movements and clicks. On the evaluation side, the form may be a string of characters, an icon or shape, or an auditory signal. The idea is to reduce the number of arbitrary relationships between the physical forms and the expressions' meanings (Hutchins et al., 1986). Using the example above, an articulatory direct execution for scheduling a crew would be to "grab" the crew icon with the mouse and place it on a graphically presented scheduling board. An articulatory direct evaluation would occur if that same crew icon now appeared on the schedule at the correct time. Forming an action specification is the activity that spans articulatory distance. For evaluation, interpretation spans the articulatory distance.

The provided articulateness of a system is closely tied to the technology available. Simple keyboards and small, low resolution screens limit the form and structure of the input and output forms, respectively. A mouse, for example, is spatio-mimetic, meaning it can provide articulatory direct input for tasks that can be represented spatially. Pictographs and icons are examples of output forms which are related to their meanings (Hutchins et al., 1986).

Norman notes that with practice and experience, crossing the "gulfs" can become easier but this does not mean that the distances have been reduced. Instead, the distances have been bridged by the users -- not the system. This implies a need to distinguish between a feeling of directness which originates from close semantic coupling between intentions and the interface language, and that which originates from practice. We believe that symptoms resulting from semantic and articulatory distance could best be seen during the learning stages of HCI, before the distance/gulf is bridged by experience.

1.5 GOALS OF THIS RESEARCH EFFORT

We have discussed high-level concepts that are important to the usability of graphical user interfaces. In order to effectively evaluate such interfaces, we need to extract information which indicates the presence of the various types of distances. Traditionally-used human factors/usability measures alone are not sufficient to adequately address these issues or deal with graphical user interfaces. Traditional measures consist of high-level summary measures such as task completion time, number and percent of errors, command frequency, percent of task complete, frequency of referencing documentation, etc. which are global and do not provide enough granularity or insight into mental processes to identify specific USI problems or their causes. They are of little value for redesign efforts.

HCI researchers have also developed a low-level class of measures which may be classified as USI-specific, task independent measures. Examples of these are input device measures such as mouse movement distance, screen-layout measures for assessing screen complexity (Tullis, 1984), legibility measures such as reading rate, and measures for calculating optimal depth vs. breadth for menu hierarchies. These measures focus on a very narrow aspect of the user interface and are valuable for fine tuning but not overall diagnostics.

The goal of this project was to develop a new class of measures and a corresponding methodology which assess the directness of the engagements supported by the USI. We call this class "measures of the directness of engagements" (table 1). These measures would build on the concepts of distances recently discussed. The measures would be cognitive as a result of the mental nature of the tasks the interface supports, and theory-based which helps direct the evaluation metrics and keeps them generic.

Table 1. Examples of Three Classes of USI Effectiveness Measures

Class	*Concepts/Measures*
Traditional behavioral performance measures	Correctness of decision Optimization of resources Task time Number of errors
Measures of the directness of engagements	Semantic distance of execution Semantic distance of evaluation Articulatory distance of execution Articulatory distance of evaluation Error analysis by stages
USI-specific, task independent measures	Menu design Mouse/keyboard measures Display legibility Screen complexity

1.6 SUMMARY

A rationale for the need for a new class of measures of user-system interface effectiveness was presented. To effectively evaluate graphical, direct-manipulation style interfaces, measures which are both cognitive and theory-based are required. The measures need to reflect concepts important to the usability of these types of systems. This includes directness in the areas of semantic and articulatory distance of execution and evaluation. In the next section, we further develop these concepts by deriving operational definitions and indicators based on these concepts, and describe a methodology and data collection techniques needed to support generation of the developed indicators.

SECTION 2

A METHOD FOR MEASURING DIRECTNESS OF ENGAGEMENTS

In this section, a method for evaluating graphical, direct-manipulation style interfaces is presented. A method is needed which describes directness in enough detail to provide indications on how to enhance the design, but not too much detail to loose sight of the overall application. We took the inverse approach, rather than attempting to measure every engagement to assess its directness, we attempted to develop indicators which reflect ***indirectness*** in any of the stages of user activity.

In this section, we discuss our derivation of indicators of semantic and articulatory indirectness based on Norman's model and Hutchinson's et al. elaboration on the model. The concept of a usability study and data collection techniques is discussed next, followed by the development of a new encoding scheme, and analysis techniques for the extraction of USI effectiveness indicators of indirectness. The following section will give an example of applying the method to evaluate a system.

2.1 DERIVATION OF OPERATIONAL DEFINITIONS AND INDICATORS FROM THE MODEL

The concepts on directness presented in section 1 are a useful starting point for assessing the usability of graphical interfaces, but further definition of the concepts is required before they can be used in practice. Our first decision was to concentrate on detecting when indirectness or large distances exist in an interface. It is easier and more useful to detect problems in directness than to quantify the directness of every HCI engagement. The concepts we are trying to measure are somewhat subjective and still require interpretation but it is possible to flag and identify potential problem areas.

We derived definitions and related indicators to address the different concepts of directness. These are shown below in Tables 2-4. The indicators were developed based on our experiences and some initial pilot testing. For semantic and articulatory indirectness of execution and evaluation, we defined potential contributors to each type of user interface indirectness (e.g., semantic indirectness exists if ...), and for each cause we provided an initial list of observable indicators/resulting user behaviors which would occur if the problem was present (tables 2 and 3). For instance, semantic indirectness would occur if the user interface had a missing high-level object that the user expected to be there. Indicators that this problem was present would be repetitious actions performed on lower-level objects. In addition to semantic and articulatory problems, there can also be problems which occur during the stages of execution and perception. For instance, if a display output is difficult to perceive, indicators would be the user performing frequent actions to improve the perceptibility of the display or making errors in perception (table 4).

Tables 2-4 refer to *expressions*, *steps*, and *actions*. We define expression as the total command sequence required to fulfill an intention. A step is a logical computer grouping of actions. Actions are the lowest level of input and their definition is variable depending on what you want to learn. For example, a step may be the sequence of actions required to move a word in a document, and the actions may include selecting the word, dragging the word, and dropping the word in a new location.

Table 2. Causes of Semantic Indirectness of Execution and Evaluation and the Corresponding Observable Indicators

Semantic indirectness of execution if:	***Indicator***
User intention not supported	• User states the desire to perform the missing function • Attempting to execute unsupported function, forced to abort
Missing high-level object	• Same step or set of actions repeated on lower-level objects
Complex expression required to accomplish intention	• Many steps/actions required to complete intention • Errors in step order • Incomplete/aborts in intentions
Semantic indirectness of evaluation if:	***Indicator***
Extra step(s) required to perform an evaluation	• Number and purpose of steps performed (e.g., to get information, or "check" something)
Difficult or impossible to perform an evaluation	• Frequency and types of evaluation errors • Evaluation not made

Table 3. Causes of Articulatory Indirectness of Execution and Evaluation and the Corresponding Observable Indicators

Articulatory indirectness of execution if:	***Indicator***
Complex steps	• Number of actions needed to perform a single step • Errors in performing the step - sequence of actions is incorrect - omit action in step - add extra action to step • Aborted step
Poor match between single action meaning and its form	• False action match • Difficulty locating/identifying action
Articulatory indirectness of evaluation if:	***Indicator***
Complex display output to interpret	• Steps or actions needed to perform a complete interpretation • Error and frequency of errors in interpretation
Poor match between display output form and its meaning	• Frequency and types of errors in interpretation

Table 4. Other Indicators Reflecting Manual or Perceptual Difficulties

Problem:	***Indicator***
Poor perceptibility of information on the display	• Frequency of steps/actions performed to improve perception • Frequency and types of perceptual errors
Poor manual interaction with the system	• Frequency of steps/actions performed to make manual interactions easier to accomplish • Frequency and type of manual/execution errors
Ease of error recovery	• Was error recovered from • Time between error and error recovery

2.2 DATA COLLECTION TECHNIQUES AND MANIPULATIONS NEEDED TO SUPPORT THE INDICATORS

The technique best suited for extracting the above indicators during an evaluation is a usability study. The indicators are extremely dependent on understanding how real users think about tasks and the ease with which they can use the computer system. Usability studies are a form of user-based evaluation which can be used to evaluate user interactions with a computer system (see MTR's 92B0000047 vol. 1 and 2 for discussions of other techniques). Usability studies (figure 4) involve collecting objective and/or subjective data on users interacting with a system or prototype. Such studies are designed to evaluate the quality of particular products or prototypes, and to improve and perfect product design, with the understanding of human behavior a useful and important by-product of the evaluation procedures (Holleran, 1991).

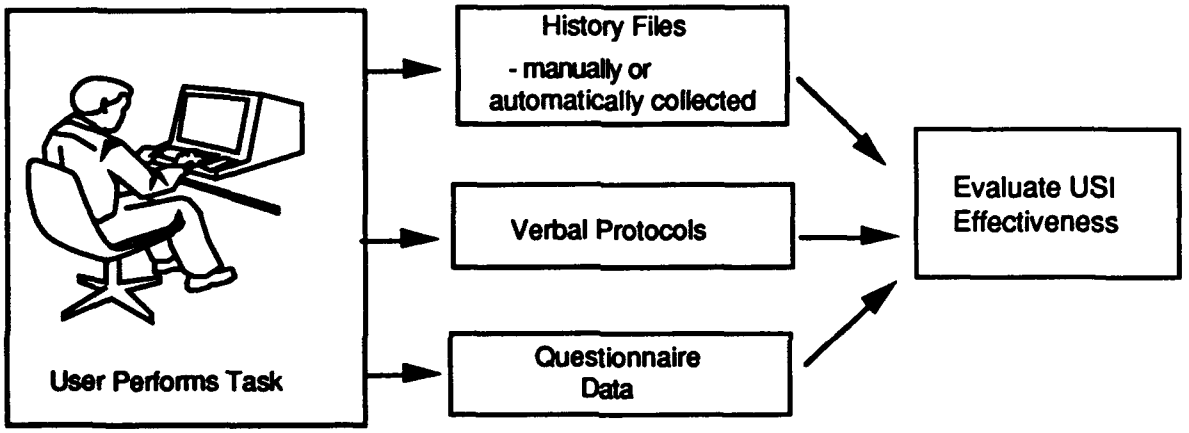

Figure 4. User-Based Evaluation

When collecting data during a usability-style USI evaluation, the purpose is to get as complete a description of the user's interaction with the system as possible. A complete description of what was done and as much information on *why* it was done is desirable. There are essentially four primary data collecti· ı means for human-computer interaction:

- video recordings of the display and users' gross activities,
- a history log of keystroke and mouse input data, which is often time-stamped,
- thoughts voiced aloud by the subjects, and
- questionnaires.

None of these methods alone, however, provides enough information to determine the degree to which the user interface supports directness of engagements.

Video recordings of the display show us the results of the user inputs. The videotape record of a user-system interaction is a valuable technique for identifying usability problems. The advantage of videotape is that it provides a complete, continuous, and real time record of the behavior of both the user and the system (Prasse, 1990). Videotape preserves the content, sequence and timing of actions that occur in the user-system interaction. These records can then be re-examined during the data analysis phase of usability testing.

History logs of keystroke and mouse input data provide us with a clear picture of ***what*** users did and ***when*** they did it. Software can be augmented to capture all user actions including keyboard entries, mouse movements, menu selections, button presses and icon manipulations. A portion of a sample keystroke file is shown below.

Line #	Time	Elapsed Time	User input
001	11:32:39	000	Pressed Button on View Button in Main Menu Bar
002	11:32:41	002	Released Button on Date Button in View Menu
003	11:32:43	002	Pressed Button on Cancel Button in Time Dialog
004	11:32:45	002	Pressed Button on View Button in Main Menu Bar
005	11:32:47	002	Released Button on Change Layout Button in View Menu
006	11:32:52	005	Pressed Button on Undisplayed SUA List in General Layout Dialog
007	11:32:58	006	Pressed Button on Add Button in General Layout Dialog
008	11:32:59	001	Pressed Button on Undisplayed SUA List in General Layout Dialog
009	11:33:01	002	Pressed Button on Add Button in General Layout Dialog
010	11:33:02	001	Pressed Button on OK Button in General Layout Dialog

The history files provide a record of *what* the user did. It alone is not sufficient to extract all the measures of interest because you can not assess what the user was *trying* to do. From the excerpt above, for instance, we can see what the user did. The user selected a menu command which opened a dialog box and then closed the dialog box. Without knowing what the user intended to do, we can not assess if this sequence of steps was the correct one to accomplish her intention. If, for example, a user opens a dialog box and immediately cancels out of it, the user could have opened it and realized it was the wrong dialog box, or the user may have intentionally opened it to obtain some information from it. Without knowing the intention, it can not be determined if a sequence was intentional or an error. In addition to the ambiguity, with history logs alone, information on the users' stages of perception, interpretation and evaluation is not available.

History logs or files are useful, however, in that they provide a detailed description of all user inputs which would be difficult to get from the video tape alone. Also, there are some indicators of USI problems that can be extracted from history files. For instance, repetitive sequences of actions could be an indicator that the user is not able to apply an action to many objects or a high-enough level object at once. So object-level information, or the need for

grouping functions, a semantic measure of execution, could be extractable. You can also extract some indicators of execute problems such as information on typos, backspaces, and frequency and duration of dragging of items. Some articulatory distance indicators are available such as menu search activity (e.g., menu, menu, menu) which implies a search for a command, performing actions in the wrong sequence, omitting actions in a sequence and similar types of error. Finally, timing between input event information is available. This type of information is not useful when task or decision times are not critical system drivers. Also, system timing may be misleading when the system being evaluated is a prototype.

The audio recordings of users asked to "think aloud" during task performance provide us with some understanding of ***why*** users did what they did. This last technique brings us closest to understanding the cognitive aspects of HCI including what the user's goals and intentions were, and their assessment of the result of their actions.

We need to understand the user's intention as well as the steps used to obtain the intention in order to assess the directness and determine the success of the following stages. The information collected in the history file needs to be integrated with a description of the user's intention and goals, as well as their interpretation and evaluation of the proceedings. The data is, however, collected in different formats and is typically not synchronized in time across formats. Thus, integrating and summarizing the data poses a challenge. Below is an example of an integrated history/transcribed-protocols data file:

"allright. Okay, so I want to see that week."
001 11:32:39 000 Pressed Button on View Button in Main Menu Bar
002 11:32:41 002 Released Button on Date Button in View Menu
003 11:32:43 002 Pressed Button on Cancel Button in Time Dialog
"Well, I probably need airspaces up there first."
004 11:32:45 002 Pressed Button on View Button in Main Menu Bar
005 11:32:47 002 Released Button on Change Layout Button in View Menu
"Who am I again? Phoenix"
006 11:32:52 005 Pressed Button on Undisplayed SUA List in General Layout Dialog
"Ah, Yankee 1."
007 11:32:58 006 Pressed Button on Add Button in General Layout Dialog
008 11:32:59 001 Pressed Button on Undisplayed SUA List in General Layout Dialog
"Ah, Yankee 2."
009 11:33:01 002 Pressed Button on Add Button in General Layout Dialog
010 11:33:02 001 Pressed Button on OK Button in General Layout Dialog

The integrated file adds a wealth of information to the history file and a previously ambiguous event now becomes clear to the data analyst. The user opened a dialog box, and then canceled it because she thought she needed to perform another step first. The user started to perform a sequence of actions to fulfill an intention, made an evaluation of whether this was a correct sequence of steps, decided it was not, aborted the current intention, and started a new sequence of actions. This turns out to be an error in evaluation because the originally planned

sequence of steps would have accomplished the intention. The computer design was not at fault in this situation because the action would have been disabled if it could not have been done at that point in time. Combining the protocols and the history file allows a more complete description of the human-computer interaction process. We can now determine the user intention, identify errors in the stages such as action specification, and obtain information on the two evaluation stages. It is interesting to note that what we have classified as an evaluation error would not have typically been recorded in a traditional error log as a user input error was not made.

With the data in the current form, it is still a lot work to extract the explanation of events we just described. Frequency of errors and comparing sequences of events across subjects would be hard to do as all quantitative measures would need to be calculated manually. What is required is a scheme for encoding the data that makes the scenario we just described very explicit. Such a scheme is proposed below.

2.3 ENCODING SCHEME

A key contribution of this research effort involved the development of an encoding scheme to structure the data collected during user-based evaluations in a way which permits easy evaluation of the directness of user-system interactions. After many iterations, an encoding scheme which seemed to usefully structure the data collected on the user-system interaction process was developed; it is loosely based on Norman's stages of user-activity model. The scheme is hierarchical in nature and, for the execution stages, somewhat similar to the Goals, Operators, Methods, and Selection techniques (GOMS) approach developed by Card et al., 1983. GOMS is a hierarchical method of analyzing the sequence of activities required by an interface for performing various tasks with the system.

Deciding what we want to learn from our usability studies was the first step in the encoding scheme's derivation. The primary purpose of a usability study is to address how well the computer system supports the needs of a user during task performance. So far, we have discussed measuring the directness of engagements-level of performance. We certainly wanted to capture information necessary for this type of analysis. As was discussed in section 1, there are also some higher level, more traditional measures of human performance which address how well the user performed the task overall. These include measures of the correctness of decisions, task time, task completion success, quality of the generated output, etc. Some of these measures are affected by the computer design but some rest with the task skills and experience of the human. For example, a very poor computer design could affect overall task time and product quality. However, a user inexperienced with the task could also have a high task time and poor product quality even if using a well-designed system. A user can interact with the computer perfectly, in terms of making no HCI errors, but still fail to achieve task objectives. Collecting information on how the human approaches a task, their problem solving strategy, their skill at the task level etc. is necessary for drawing overall conclusions about the computer system effectiveness. We wanted to extract these measures, in addition to measures on the directness of engagements, from our encoded data.

Using the stages of user activity model as a basis, our first idea for an encoding process was to use every user intention as a starting point and supply an encoding for all of the following stages from the model. This brute force approach quickly runs into trouble in two ways. First, adding the perception, interpretation, and evaluation stage to every user intention greatly increases the amount of information, and, for the most part, these stages will just be coded as having been performed correctly. Thus, only when there is a problem in these latter stages are they of real interest. Also, the process of perception is continuous, and difficult to code as a discrete action. Only an error in perception or taking extra actions to improve perception is observable. The second problem is identifying the appropriate level of intention to encode.

At what level of interaction do we apply the cycle? As noted by Norman (1986), the HCI process can be broken down further and further until the level of a user's intention would be to move the cursor. As the cursor is moving, there would be a continuous perceptual activity. The completion of the cursor movement can then be interpreted and evaluated. When working at such a low level, a single cursor movement activity is broken down into many stages, and actually creates more data with very little information added. Other evaluation techniques such as GOMS (Card et al., 1983) and the cognitive walkthrough technique (Lewis et al., 1990) have the same problem. Activities can be broken down lower and lower until the information may be below the level of interest. The best level to work at depends on the questions to be answered by the study. For our purpose, which is evaluating new prototyped software systems, we are not interested in the very lowest keystroke level of interaction. We want to focus on the semantic and articulatory levels rather than on the detailed execute level. With that constraint in mind, a two-level encoding scheme was created -- one level focusing on the user's cognitive strategies or semantic level, and the second on the articulatory/execution level. The second level was implemented at a user-interface object level, rather than at an individual keystroke, cursor movement level.

2.3.1 Semantic Level

When trying to identify the stages associated with human-computer interaction activities, the first four stages of the user activity model (goal, intention, action specification, and execute) are easier to identify than the three stages spanning the gulf of evaluation. Determining the user's goals and intentions is relatively straightforward, particularly if predetermined tasks are being performed. Information on action specification can be inferred from the observed executed actions, although the mental processes involved in formulating the sequence of actions are not observable. Examining sequential data records reveals information on whether users think they have accomplished their intention, the evaluation stage.

Based on these concepts, and the more traditional task measures of interest, the following encodings were developed at the semantic level of human-computer interaction. Each user goal corresponds to an overall objective, probably provided by the conductors of the usability study. The remaining codes allow us to categorize and identify the user's objectives.

Within each goal, we wanted to identify the task-space intentions the users have to accomplish the goal, their intentions to execute specific computer steps to accomplish a single task, and intentions they form to improve the perceptibility of the computer workspace. Each of these user intentions is completed with a corresponding evaluation code that contains information on the success of their endeavors. Also included in this semantic-encoding level are errors which are made in any of the stages of user activity. From this scheme, information on the user's goals, intentions, problem solving strategies, computer method strategies, goal achievement, errors in each stage of user activity, and whether or not an error was recovered from can be obtained. Successfully applying the codes involves extracting and analyzing all the various types of data collected. Each code is defined more fully below.

Goal — Achievement to be obtained by the user, usually predetermined by the experimenter. The items to be accomplished in a task scenario.

Task intention — The intention to complete one task contributing to the completion of a goal. This is still in the user task space. For example, a goal may be to schedule all requested missions, but a task intention may be to schedule a particular mission.

Intention to execute — A task step, to accomplish the given task intention. Each task intention is performed by a single or a sequence of intention to executes. The intention to execute is the description of the step the user wants to perform. For example, opening a dialog box, performing some functions in the box, and closing the box often characterized a single task step. The purpose for executing that step is the intention to execute. One or more such steps may be required to perform a single task intention. In the semantic encoding level, the intention to execute was not further broken down. The details of the individual actions in the step are provided in the articulatory level.

Intention to improve perception — Task intentions were characterized by their intent to move users closer to their goal, to the completion of a particular task. However, many direct manipulation systems are graphical in nature and an artifact of using them is that the user sometimes is forced to take one or more steps to improve the perceptibility of the work area. This is distinguishable from the task intention in that it is necessary but does not directly move the users closer to their goal, nor is it done with the intent to complete a task. It is on the same level as task intentions because they interrupt the completion of a goal and have their own set of intentions to execute. Finally, it is an artifact that the system design should minimize and is therefore identified separately from other task intentions.

Evaluation — Each intention for either task, execution, or perception is closed off with a corresponding evaluate state which encodes our interpretation of the success of the endeavor. This state can be coded as:

"OK", meaning proper completion of the corresponding goal, task, or execute;

"Inc", meaning the user has not fully completed the corresponding intention (e.g., changed the time correctly but forgot to convert it to EST units);

"abort", meaning the user has abandoned the corresponding goal, task, or execute; or

"wrong", the series of actions chosen to accomplish the task or execute were not the right sequence of actions.

Errors detected at any stage were included in the semantic level encodings. They are defined as follows.

Error in intention — Occurs when the user is making the classically-defined "mistake". The intent of what they want to do will not move them closer to their goal, even if they execute the intent correctly. This type of error is often independent of the user-interface, and may be the result of misunderstanding the task, or forgetting some of the task details. For example, if a user was asked to create a schedule for the time period 8 - 12 August, and they instead create it for the time period 8 - 12 September because they thought September was the correct month, they have made an error in intention. Note that if they meant to schedule for August but inadvertently selected September rather than August from a list of months, this would be an execution error, and if they failed to notice the error, an error in evaluation as well. This highlights the fact that the type of error can not be ascertained from the history log of user inputs alone; an understanding of the user's thought process is also required.

Error in action specification — Occurs if the user performs an action or series of actions on the computer which are not correct to accomplish the intended step. This can range from having to search through menus to locate a command, to opening the wrong dialog box, to omitting actions in a sequence of actions, etc.

Error in execution — Occurs if the manual interaction with the computer is not as intended. These types of errors include typos, selecting an item adjacent to the intended item, etc.

Error in perception — Occurs if the user encounters difficulty thought to be caused by a break-down in the perceptual processing of the display. For instance, the user schedules the wrong mission because the mission icons were small and close together, or the status indicators were too small to be read correctly.

Error in interpretation	Occurs if the user perceives the information correctly but fails to interpret the system state correctly. For example, if a mission icon turns red and the user determines this to mean "scheduled and OK" when in fact it means "scheduled and in conflict with another mission", an error in interpretation has occurred.
Error in evaluation	Occurs when the user performs the evaluation stage, determining whether s/he moved closer to his/her goal, and either they think they did move closer when they did not, or vice versa. An example would be a user starts to perform the correct sequence of actions in an execute, thinks he needs to do some other action first to achieve his goal when he does not, and abandons his set of steps to do an unrequired sequence of actions first.
Error recovery	Occurs when a user notices that s/he has made an error and corrects it. Usually, typos and other low-level errors are detected almost immediately and corrected. If an error is not noticed at the time the user leaves the task, the task is usually coded as incomplete.

The encodings are summarized in table 5.

The coding scheme at this level gives information on the user's overall strategy, the number and types of steps performed to carry out a task intention, whether the step was performed correctly at the computer level, whether the task was the right task to meet the goal, whether it was carried out completely, and where and what types of errors were made at each stage of user activity, as well as whether it was recovered from or not. Information on the actual sequence of commands and user inputs selected are not included at this level but at the articulatory level. We believe the semantic level is generic enough to be applied to most applications with direct-manipulation style interfaces.

Table 5. Semantic-Level Encodings

Encoding	*Definition*
Goal	Scenario step.
Task intention (Int.task)	An intention to complete one task contributing to the completion of a goal.
Perception intention (Int.per)	An intention to improve the perceptibility of a display.
Intention to execute (Int.exe)	One computer step (may be comprised of multiple actions) leading to the completion of a task intention. Several steps may be required per task intention.
Evaluate (Eval)	The success with which the intention was accomplished.
Error in intention (Err.int)	The intention was incorrect and will not accomplish the goal.
Error in action specification (Err.acsp)	Wrong sequence of actions to accomplish the intention to execute.
Error in execution (Err. exec)	Manual, motor error in executing.
Error in perception (Err. per)	Break-down in human perceptual processing of information on a display.
Error in interpretation (Err. inter)	User fails to interpret system state correctly.
Error in evaluation (Err.eval)	User mistakenly thinks has or has not moved closer to the goal.
Recovered error (Rec.err)	Error was detected and recovered from.

2.3.2 Articulatory Level

The articulatory encoding level focuses on the actual sequences of commands and user inputs required to perform each intention to execute; the encodings are hierarchical and the articulatory level is the lowest level. Execution could be a single keystroke or a mouse movement series of actions, depending again on the level that the analyst wishes to break down the data. We attempted to keep the recorded action at the user-interface object level. Thus, actions encoded were the use or selection of: menus, commands, buttons, window manipulations, selection from lists of items, data-entry field actions, and scroll bar actions.

These are generic, user-interface objects. Selecting a command could, for example, result in an action being taken, such as approving a mission, or it could result in the opening of a dialog box. For each, the actual menu, command, or button name is recorded. The manipulation of some application-specific objects could also be of interest, and the encodings should be added on an individual system basis. The articulatory encodings are shown in table 6.

Table 6. Articulatory-Level Encodings

Encoding	*Definition*
Menu	A menu was opened
Command	A command was selected
List-Select	An item is selected from a list
Button	A button was selected
Field	An action was taken in a field
Scroll	A scroll bar action was performed
Window	A window action was performed
Application-specific objects	Encodings to track the manipulation of application-specific objects

An example is given below in figure 5 to illustrate the hierarchical nature of the encodings for both the semantic and articulatory levels, illustrating the encoding relationships.

2.4 EXTRACTING USI INDICATORS FROM THE ENCODED DATA

This encoding scheme provides a generic structure to be applied to integrated sequential data. Since analysis and interpretation of the various sequential data streams is needed to apply the encodings, it is itself a form of analysis and it provides more information than any one type of sequential data alone or the uncoded integrated data file alone. Additionally, the data is now in a form amenable to quantitative analysis. From the encoded data, much information on the effectiveness of the human-computer interaction process can be extracted.

Goal 1

Task intention (task A)
Intention to execute (step 1)
Button 1
Execute error
Button 2
Recover from error
Evaluate step 1

Intention to execute (step 2)
Menu
Command
Error in action specification
Menu
Command
Recover from error
Button
Evaluate step 2
Evaluate task A

Perceptual intention (task C)
Intention to execute (step 1)
Timebar
Timebar
Evaluate step 1
Evaluate perception

Evaluate Goal 1

Figure 5. Conceptualization of the Encoding Hierarchy

At the highest level, the users' problem solving strategies can be seen as well as the order in which they attempted to conduct each goal, their task and perceptual intentions, the order of their intentions, the method selected to accomplish each intention, the success of the various levels of endeavors, whether they successfully completed each goal of those assigned and where and what types of errors were made. At the lower, articulatory level, the actual actions the user selected to accomplish each task and perceptual intention, the number of actions per intention, the order in which they were taken, whether an execute error was made, etc. can be seen.

The indicators of USI indirectness (given previously in tables 2-4, section 1) can now be extracted from the encodings. The indicators for semantic indirectness and how they can be identified are discussed below.

Semantic Indirectness of Execution

- **User intention not supported**

Semantic indirectness exists if a user has an intention which the computer does not support at all. If the user knows the intention is not supported by the system, s/he may voice the desire aloud. This information can be extracted from the protocols. For instance, a user may say "I wish I could bring up all parts of this mission on the screen at once." There is no function which supports this desire but the request for it is now known and can be evaluated as to whether it should be added or not.

A second indicator of the same problem is aborting a particular attempt to execute an unsupported intention; in this case, the user does not know the intention is not supported. This would be coded as a task intention, for example, Task Intention (show all parts of mission A), followed by an intention to execute and a string of actions. The user will not be able to successfully accomplish his intention, so the corresponding evaluates will be coded as aborts.

- **Missing high-level object**

A case of semantic indirectness occurs when a task can be performed but not at the level desired. This results in applying an action to a series of lower-level objects repetitively. For instance, a user may wish to change all four parts of a mission from 9:00 to 8:00 at once. Since this is not supported with the system, a series of four Task Executes applied to each part of the mission will be required to accomplish the intention to change the times of that mission. Thus, an indicator of this condition is that the same steps will occur repeatedly to accomplish a single task intention. Note the actual actions used to accomplish each execution may or may not be identical as there may be alternative methods of executing a time change.

- **Complex expression required**

Complexity of the expression required to accomplish an intention is a cause of semantic indirectness. If many steps (intentions to execute) are required, the correct steps are not obvious, or the order of steps is not immediately apparent, the expression could be labeled complex. The indicators of this condition are: the number of steps per task or perceptual intention, steps are missing or extra steps are performed, and steps are performed in the wrong sequence. Other indicators would be task or perceptual intentions are aborted, incomplete, or wrong. This information is now easily extracted from the encoded data since we know what the user intention was, we have all the information on the actual sequence of steps performed to execute the intention, and the task intention evaluation provides data on the success of the endeavor.

Semantic Indirectness of Evaluation

- **Extra step(s) required to perform an evaluation**

There are two indicators that semantic indirectness of evaluation exists; this type of indirectness is usually attributable to lack of or poor feedback in a system. One indicator involves having to perform steps (intentions to execute) such as "check whether mission A is now scheduled for 8:00" in addition to the expected executes required for a task. This indicates that the user needed additional information to perform the evaluation of whether the intention was successfully accomplished.

- **Difficult or impossible to perform an evaluation**

Semantic indirectness of evaluation also occurs if it is very difficult or impossible for the user to perform an evaluation. For very poor designs, a user may be totally unable to ascertain whether progress was made toward the execution of an intention. This inability to perform an evaluation is usually due to a lack of feedback on the part of the system. Indicators of this condition would be that an evaluation is not made (extract from the protocols), and a high frequency of evaluation errors.

There are also causes of articulatory indirectness of execution. This is affected by whether the form of the input action matches the intent of the action. Since there is often more than one action required to carry out a single step, the ease of identifying the correct order of the actions is also important. Indicators of articulatory distance are discussed below.

Articulatory Indirectness of Execution

- **Complex steps required**

Complexity of a single step expression is a cause of articulatory indirectness. Indicators of indirectness would be a large number of actions per intention to execute. If, for example, changing the mission time of a single part of Mission A from 9:00 to 8:00 required 12 actions, this would be an indicator of indirectness. An example of a direct execution would be dragging the mission icon displayed on a schedule from the 9:00 time slot to the 8:00 time slot with one action. Obviously there is some subjectiveness in how many actions are too many. The frequency with which each of the steps occurs will be an important factor in determining indirectness. Frequently performed steps should have the fewest actions for proper execution (unless the step frequency is due to a semantic distance problem, then the problem should be fixed there). Other indicators of step complexity are errors such as performing actions in the incorrect sequence, omitting actions in a sequence, and adding extra, unnecessary actions. The number of actions per intention to execute indicator is easily obtainable from our encoded data. The latter would be coded as errors in action specification.

- **Poor match between a single action meaning and its form**

Another cause of articulatory indirectness of execution is a poor match between the form and meaning for a single action. This can be indicated by errors in action specification of the false action match type (Lewis et al., 1990), where the wrong form seems to match the user's intent so the user selects the wrong action. Another indicator of a poor match would be the inability to find or identify the right command in a set of menus or if the user has difficulty in locating the correct action.

Articulatory Indirectness of Evaluation

- **Complex display output**

A complex display or displays with a poor layout are causes of articulatory indirectness of evaluation. An inability to easily interpret the system state, or having to perform many actions or absorb a lot of information to interpret the system state are indicators that this state exists. This can be determined from protocols, and the frequency of errors in interpretation, and possibly from the sequence of actions performed during interpretation. Interpretation is difficult to assess as it involves the transfer of information from the display to the user. It is difficult, without special data collection equipment such as an eye-tracker, to assess the number of pieces of information used or the order in which information was assessed for an interpretation.

- **Poor match between display output form and its meaning**

Another cause of articulatory indirectness of evaluation is a poor match between a display output form and its meaning. This will result in errors in interpretation, and actions may be taken to get more information to aid in the interpretation. For example, having to refer frequently to a legend or key to interpret icon meanings is an indicator that they are not directly conveying their meaning.

Indicators reflecting manual or perceptual difficulty

Indicators of perceptual and manual difficulty also exist. They are described below.

- **Poor manual interaction with the system**

The ease of physically interacting with a system is a function of the input devices and their parameters, the fit of the input device capabilities to the task, the manual dexterity of the user, the limitations of the display real estate, etc. The best indicator of manual problems of execution are frequent errors in execution. Some of the same actions which are taken to improve perception (see below) also improve the ease of the manual interaction because objects are made larger. Therefore, actions taken to increase the size of objects which are selectable could be an indicator of manual interaction difficulties.

- **Poor perceptibility of information on the display**

The frequency of intentions to improve perception or the need to adjust the workspace are indicators of poor user perception. Having to improve the perceptibility of the workspace is a function of the task, the size and resolution of the display, the design of the user interface, the design of a particular display, and even the user strategy selected to perform a task with a given system. It may be unfeasible to totally eliminate this type of function with current technology but computer aids, innovative design, or encouraging particular strategies may help to minimize it. Oftentimes, there is a trade-off between minimizing this type of action, the perceptibility of information on a display, and the amount of context available. When viewing large amounts of information, context is increased, movement is minimized but perceptibility is reduced. Very frequent intentions to improve perception indicate there may be a problem.

A method for recording these indicators is presented below.

2.5 PROCESS FOR IDENTIFYING AND RECORDING INDICATORS

To efficiently apply this USI evaluation method, the indicators extracted from the encoded data need to be recorded. Additionally, to assess whether USI design changes are called for, information is required not only on each user's performance but also on the number of users who experience problems in the same USI area, and the similarity of their problems. To address these issues, the following data summarization table was developed. This table serves several purposes. First, it aids in extracting the indicators from the encodings for each user, particularly number of steps per task intention, number of actions per step, reason for each intention, and the frequency and type of task intentions and intentions to execute. It also summarizes the number and types of errors made during each intention, and the evaluation of the success of each intention is readily apparent.

Int.task	Freq	Int.exe (step)	Freq	# actions per int.exec	Eval of int.exec	Eval of int.task	Errors	Comments

The tables also summarize each user's performance for comparison across subjects. For each goal, the usability study conductor can assess whether many or only one user had difficulties with certain goals, and whether their problems were similar or different.

One disadvantage of the table is that the sequencing of events is lost. This information, however, is available in the encoded files. The comments column can be used to record indicators of interest which are not inherent in the format, such as the user's stated desires obtained from protocols, user's comments on the USI, more complete descriptions of errors, noting of low-level repetitious actions in the data, etc.

Finally, to efficiently apply this method a determination needs to be made of whether an interface design change is required to fix a problem or whether the problem is due to user inexperience with the system. For example, if indicators show a user is performing a task inefficiently, it could either be due to indirectness in the interface or the user may be unaware of a more efficient method. While we would like all systems to be user-centered and every interaction to be intuitively obvious to the user without any training, this is not likely to be the case for the near future. With our evaluation method the ability to easily compare user's performance at an interaction level allows assessments of whether problems are specific to a single user or are occurring across users. Each situation requires interpretation on the part of the usability analyst. User's performance changes, as they learn and become more experienced with the system, can also be tracked with this method.

2.6 SUMMARY

Indicators that various types of semantic and articulatory indirectness are present in a system were derived. By collecting and integrating both protocols and history files a fairly complete description of the human-computer interaction process is provided and identification of the indicators of indirectness can be fully supported. To enable extraction and quantification of the indicators, an encoding scheme was developed. Finally, a data summarization table was developed for recording metrics from the encoded data. The table allows for ease of comparison of indicators across users. With this method, we feel that a more complete graphical user interface evaluation is possible than with any of the existing, traditionally used evaluation methods. In the next section, we describe an actual usability study that we conducted and apply this methodology to analyze the collected data.

SECTION 3

APPLICATION OF THE EVALUATION METHODOLOGY TO THE MILITARY AIRSPACE MANAGEMENT SYSTEM

In this section, the evaluation methodology is applied to the data collected during a usability study performed on the Military Airspace Management System (MAMS) prototype. The steps taken to format the data for application of the encodings, the tool used to aid in applying the encoding language, a summarization of the USI measures of effectiveness extracted from the data, and USI implications are discussed.

3.1 MILITARY AIRSPACE MANAGEMENT SYSTEM (MAMS) USABILITY STUDY

A usability study was conducted on a MAMS prototype. The MAMS displays and forms are described in MTR 92B0000047 vol. 2. Only the main MAMS display is shown here, in figure 6. The MAMS usability study was selected to test this methodology because it possesses a graphical, direct-manipulation style interface. The interface was implemented with Motif.

3.1.1 Test Participants

Six test participants were used in the MAMS usability study. Four test participants were from the Air Force, one test participant was from the Navy, and one test participant was from the Marines. Five out of the six test participants had previous airspace scheduling experience and four out of the six test participants had participated in the MAMS users group, which had contributed to the system requirements. Additionally, the MAMS USI designer participated in the evaluation to provide a baseline of expert performance in terms of user interface familiarity.

3.1.2 Apparatus

The test sessions were conducted in the User-System Interface Technology Testbed (USITT) Evaluation Facility in MITRE M-Building. The MAMS prototype was hosted on a Sun workstation. The software was specially instrumented to collect time-stamped user inputs. A tripod-mounted Panasonic video camera with time and date stamps was used to videotape the display for each participant.

3.1.3 Procedure

Test participants were scheduled for a one-day test session beginning with a 90 minute training session and a pre-test questionnaire. The training session provided detailed information and a demonstration of all applicable topics. A definition of a "good" schedule versus a "bad" schedule for the purposes of the usability test was provided. The test session consisted of the

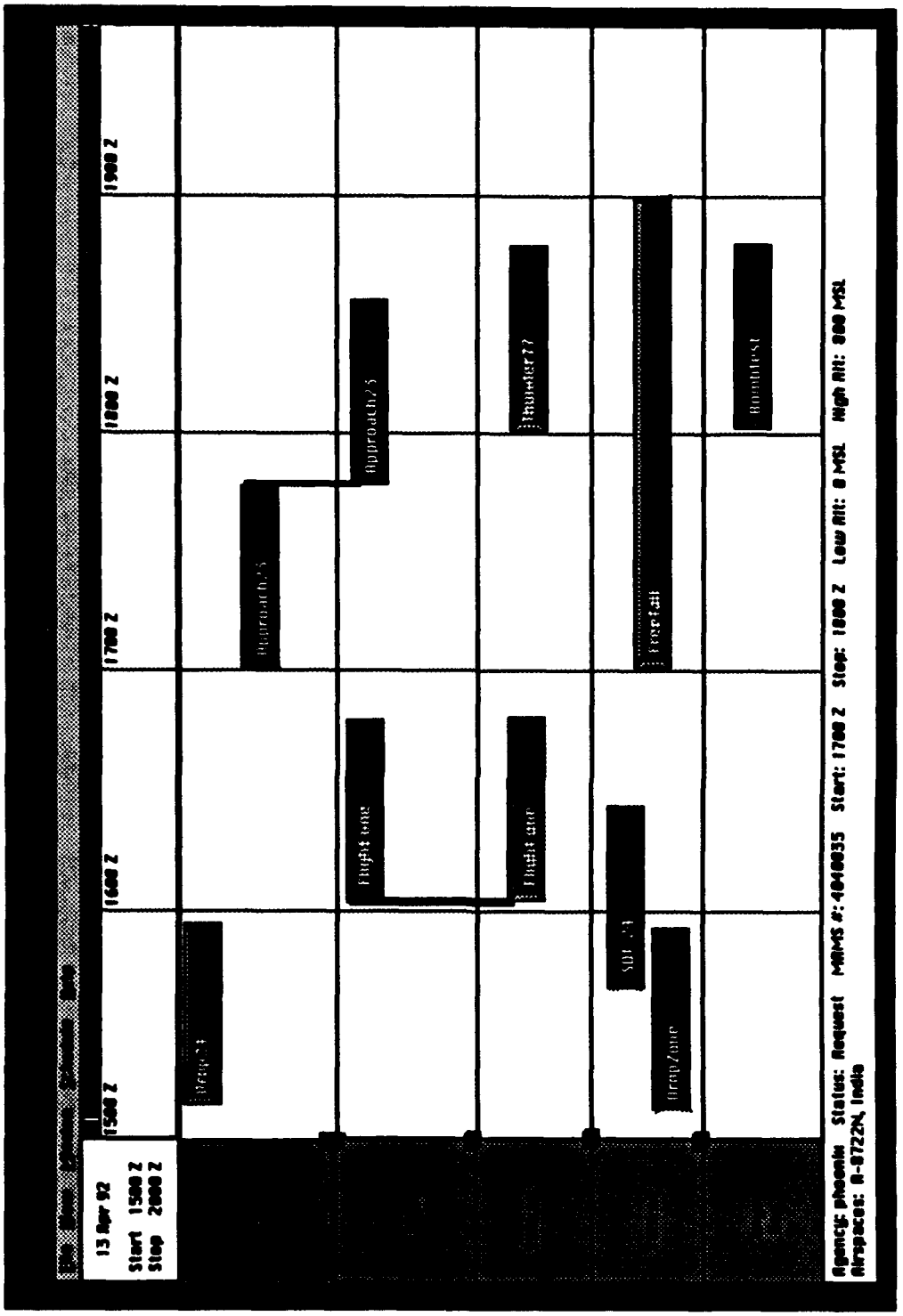

Figure 6. MAMS Main Screen

participants completing the tasks outlined in the test scenario (see Appendix A) and completing a post-test questionnaire. To keep the testing period reduced to a reasonable length of time, only a subset of the total MAMS functionality was tested. A basic scheduling scenario was developed that incorporated the core tasks of airspace scheduling. These tasks included: approving mission requests, entering missions into the schedule, resolving mission schedule conflicts, finding mission data in the schedule and generating mission reports. Additionally, test participants were asked to build groupings or folders of airspaces to be used to facilitate repeated data entry procedures. Test participants were encouraged to work at their own speed and were given breaks whenever they requested them. A pilot study had been previously conducted to verify the training and test procedures.

Test participants were provided with copies of the training materials and a MAMS prototype users guide for their use during the test session. Test participants were additionally provided with a quick reference template which presented information on keyboard accelerators, as well as examples of valid date and time entries. Test administrators were available via intercom to assist test participants when they requested help.

3.2 DATA SELECTED FOR FURTHER ANALYSIS

During some of the test sessions, software problems were encountered and the system failed; this is to be expected when using a prototype. In some instances, the keystrokes before the software failure were lost and in one case, the system failed twice and all of the data in between the failures was lost. As a result of this, and because of time constraints for completing this phase of the research, the data for the two participants with the data loss were excluded from extensive analysis. The USI design expert's data was analyzed, to provide a baseline of expert system (not task) performance. This made a total of five data files for analysis (referred to as participants 1-5). This number is consistent with current research on the number of participants required for usability testing. Virzi (1992) has found that 80% of all usability problems are detected with four or five participants, and the most severe problems are detected in the first few participants.

3.3 DATA FILTERING AND INTEGRATION

The first step in the data analysis process was to get the data in a form amenable to applying the encoding scheme. As was mentioned earlier, the process of combining history file data with protocol data is challenging and manual in nature. An excerpt of a raw history file, collected by instrumenting the MAMS software, is shown below.

User Action		Time	Object
ButtonPress	Button1	11:32:39	viewCascadeButton View
FocusIn		11:32:39	mainWindow
FocusIn		11:32:39	mainWindowForm
FocusIn		11:32:39	mainMenuBar

FocusIn		11:32:39	mainWindow
FocusIn		11:32:39	menuShell2
ButtonRelease	Button1	11:32:40	changeDatePushBu Set Date and Times.
FocusIn		11:32:40	MAMS
FocusIn		11:32:41	ScreenSetup_popu p
ButtonPress	Button1	11:32:43	screenSetupCancel
ButtonRelease	Button1	11:32:43	screenSetupCancel

The raw history file needed to be filtered to reduce the size of the file; this was done by removing extraneous data such as key releases which are preceded by a keypress. The filter also combined typed text into a string, and summarized dragging actions to eliminate the description of every location an item was dragged across. The filter also produced elapsed times which reflect the users delay between keystrokes plus the computer response time from the previous input.

The instrumented software used for this study had some limitations in the user actions that were recorded and some user inputs were not captured. Actions not recorded included the specific item name that was selected within a list, the characters that were deleted within a field, and enlarging or reducing dialog boxes to improve visibility. Double and triple mouse clicks were recorded as single clicks and the video tape was used to differentiate between them. Some inputs were ambiguous, for example, "pressed the space bar" may be indicating a typed space or the deletion of the contents of a field. Again, the video tape was used to determine the purpose of the input.

After filtering, the history file was as shown below. The data items are: line number, the time of execution, elapsed time in seconds, and the filtered user input.

001 11:32:39 000 Pressed Button on View Button in Main Menu Bar
002 11:32:41 002 Released Button on Date Button in View Menu
003 11:32:43 002 Pressed Button on Cancel Button in Date Dialog
004 11:32:45 002 Pressed Button on View Button in Main Menu Bar
005 11:32:47 002 Released Button on Change Layout Button in View Menu
006 11:32:52 005 Pressed Button on Undisplayed SUA List in General Layout Dialog
007 11:32:58 006 Pressed Button on Add Button in General Layout Dialog
008 11:32:59 001 Pressed Button on Undisplayed SUA List in General Layout Dialog
009 11:33:01 002 Pressed Button on Add Button in General Layout Dialog
010 11:33:02 001 Pressed Button on OK Button in General Layout Dialog
011 11:33:05 003 Pressed Button on View Button in Main Menu Bar
012 11:33:10 005 Pressed Button on Date Field in Date Dialog
013 11:33:13 003 Typed "133" in Start Date Field in Date Dialog
014 11:33:39 026 Pressed Button on Day Field in Date Dialog
015 11:33:40 001 Typed "7" in Day Field in Date Dialog
016 11:33:45 005 Pressed Button on OK Button in Date Dialog

To integrate the history file with the "voiced-aloud thoughts", the filtered history files were printed out, the video tapes were viewed and the verbal protocols were manually transcribed onto paper in the appropriate location on the history file. The problems experienced by the participants were also noted and recorded. The protocols were then typed into the filtered history files using the EMACS editor in UNIX. Below is an example of the integrated history/transcribed-protocols data file:

"Alright. Okay, so I want to see that week."
001 11:32:39 000 Pressed Button on View Button in Main Menu Bar
002 11:32:41 002 Released Button on Date Button in View Menu
003 11:32:43 002 Pressed Button on Cancel Button in Date Dialog
"Well, I probably need airspaces up there first."
004 11:32:45 002 Pressed Button on View Button in Main Menu Bar
005 11:32:47 002 Released Button on Change Layout Button in View Menu
"Who am I again? Phoenix"
006 11:32:52 005 Pressed Button on Undisplayed SUA List in General Layout Dialog
"Ah, Yankee 1."
007 11:32:58 006 Pressed Button on Add Button in General Layout Dialog
008 11:32:59 001 Pressed Button on Undisplayed SUA List in General Layout Dialog
"Ah, Yankee 2."
009 11:33:01 002 Pressed Button on Add Button in General Layout Dialog
010 11:33:02 001 Pressed Button on OK Button in General Layout Dialog
011 11:33:05 003 Pressed Button on View Button in Main Menu Bar
"Okay, now I want to see my dates. Start date of.."
012 11:33:10 005 Pressed Button on Date Field in Date Dialog
013 11:33:13 003 Typed "133" in Start Date Field in Date Dialog
014 11:33:39 026 Pressed Button on Day Field in Date Dialog
"For 7 days."
015 11:33:40 001 Typed "7" in Day Field in Date Dialog
"For duration of five hours I will keep, cause I like to see five hours."
016 11:33:45 005 Pressed Button on OK Button in Date Dialog
"Okay, oops, forgot to change the month."

3.4 APPLYING THE ENCODING SCHEME TO THE INTEGRATED DATA FILES

Working from hardcopies of the integrated data files, the video was viewed a second time and the encoding scheme was applied to the hardcopies of the data files. The integrated files were then transferred to the IBM PC from the Sun workstation. Next the encoding language was applied to the integrated data files using the SHAPA software tool.

3.4.1 Encoding with SHAPA

SHAPA (Software for Heuristically Aiding Protocol Analysis) was developed at the University of Illinois at Urbana-Champaign Engineering Psychology Research Laboratory (Sanderson et al., 1989). SHAPA is a protocol analysis environment where researchers can encode data with any encodings they choose. Data can be coded in a variety of different ways, depending on what questions need to be answered, and analysis can occur at many levels. SHAPA supports sequential data analysis of encoded protocol segments. The analysis techniques it supports include transition matrix analysis, lag sequential analysis, and frequency of cycles. SHAPA works on single-stream, un-timestamped verbal and non-verbal protocols; it runs on an IBM PC or compatible.

To encode the data with SHAPA, predicates (names of the codes to be applied) are specified. These would be our task intention, intention to execute, etc. as defined in tables 5 and 6. To the articulatory-level encodings, we added two application-specific codes: manipulations of the timebar (which affected how much of a schedule was viewed), and manipulations of the mission icons. The mission icons could be moved by dragging with the mouse, and selected by clicking on them. Once selected, information about that mission appears in an information field, and selected commands could then be applied to it (approving, denying, editing, etc.).

Each predicate, which is general, can also have a user-defined value, which is a specific instance or further description of the predicate. Figure 7, an excerpt of the semantic level encodings from the MAMS study, illustrates many of these concepts. For the predicate INT.TASK, 1-1-setdate was the value descriptor. 1-1 means this is the user's first task intention for goal 1. Setdate means the task was to set the date and time of the schedule. The SHAPA screen looks like the figure. The data file is displayed in the right column of the screen and the corresponding encodings are typed in the left column. SHAPA will prompt for correct syntax based on the user pre-defined predicate and value list. The encoded files could then be printed out, with an option to also print the protocols and keystrokes around the encodings. An articulatory level encoding is shown in figure 8. The complete set of predicates and values used in the MAMS study are provided in Appendix B.

	"Alright. Okay, so I want to see that week."
GOAL(1-setdate)	001 11:32:39 000 Pressed Button on View Button in Main Menu Bar
INT.TASK(1-1-setdate)	002 11:32:43 004 Pressed Button on Cancel Button in Date Dialog
INT.EXEC(1-1.1-setdate)	"Well, I probably need airspaces up there first."
ERR.EVAL (e1*1-1.1-setdate-thought needed airspaces on display)	003 11:32:45 002 Pressed Button on View Button in Main Menu Bar
EVALUATE(1-1.1-setdate-abort)	004 11:32:47 002 Released Button on Change Layout Button in View Menu
INT.EXEC(1-1.2-setlayout)	"Who am I again? Phoenix"
	005 11:32:52 005 Pressed Button on Undisplayed SUA List in General Layout Dialog
	"Ah, Yankee 1."
	006 11:32:58 006 Pressed Button on Add Button in General Layout Dialog
	007 11:32:59 001 Pressed Button on Undisplayed SUA List in General Layout Dialog
	"Ah, Yankee 2."
	008 11:33:01 002 Pressed Button on Add Button in General Layout Dialog
	009 11:33:02 001 Pressed Button on OK Button in General Layout Dialog
	010 11:33:05 003 Pressed Button on View Button in Main Menu Bar
EVALUATE(1-1.2-setlayout-ok)	"Okay, now I want to see my dates. Start date of.."
	011 11:33:10 005 Pressed Button on Date Field in Date Dialog
INT.EXEC(1-1.1-setdate)	012 11:33:13 003 Typed "133" in Start Date Field in Date Dialog
	013 11:33:39 026 Pressed Button on Day Field in Date Dialog
ERR.EXEC(e2*1-1.1-setdate-typo)	"For 7 days."
	014 11:33:40 001 Typed "7" in Day Field in Date Dialog
ERR.ACSP(e3*1-1.1-setdate-forgotsetmonth)	"For duration of five hours I will keep, cause I like to see five hours."
EVALUATE (1-1.1-setdate-inc)	015 11:33:45 005 Pressed Button on OK Button in Date Dialog
	"Okay, oops, forgot to change the month."

Figure 7. Application of Semantic Level Encodings

GOAL(1-setdate)	"Alright. Okay, so I want to see that week."
INT.TASK(1-1-setdate)	001 11:32:39 000 Pressed Button on View Button in Main Menu Bar
INT.EXEC(1-1.1-setdate)	002 11:32:43 004 Pressed Button on Cancel Button in Date Dialog
MENU(view,m)	"Well, I probably need airspaces up there first."
COMMAND(date,m)	003 11:32:45 002 Pressed Button on View Button in Main Menu Bar
BUTTON(date-cancel)	004 11:32:47 002 Released Button on Change Layout Button in View Menu
EVALUATE(1-1.1-setdate-abort)	"Who am I again? Phoenix"
INT.EXEC(1-1.2-setlayout)	005 11:32:52 005 Pressed Button on Undisplayed SUA List in General Layout Dialog
MENU(view,m)	"Ah, Yankee 1."
COMMAND(layout,m)	006 11:32:58 006 Pressed Button on Add Button in General Layout Dialog
LIST_SELECT(layout-undis)	007 11:32:59 001 Pressed Button on Undisplayed SUA List in General Layout Dialog
BUTTON(layout-add)	"Ah, Yankee 2."
LIST_SELECT(layout-undis)	008 11:33:01 002 Pressed Button on Add Button in General Layout Dialog
BUTTON(layout-add)	009 11:33:02 001 Pressed Button on OK Button in General Layout Dialog
BUTTON(layout-ok)	010 11:33:05 003 Pressed Button on View Button in Main Menu Bar
EVALUATE(1-1.2-setlayout-ok	"Okay, now I want to see my dates. Start date of.."
INT.EXEC(1-1.1-setdate)	011 11:33:10 005 Pressed Button on Date Field in Date Dialog
MENU(view,m)	012 11:33:13 003 Typed "133" in Start Date Field in Date Dialog
COMMAND(date,m)	013 11:33:39 026 Pressed Button on Day Field in Date Dialog
FIELD(date-date-edit)	"For 7 days."
FIELD(date-durdays-edit)	014 11:33:40 001 Typed "7" in Day Field in Date Dialog
BUTTON(date-ok)	"For duration of five hours I will keep, cause I like to see five hours."
EVALUATE(1-1.1-setdate-inc)	015 11:33:45 005 Pressed Button on OK Button in Date Dialog
	"Okay, oops, forgot to change the month."

Figure 8. Articulatory Level Encoding Example

3.4.2 Evaluation of the Data Transformation Process

The times required for the data transformation process are summarized below. The table shows approximate times for different parts of the data transformation process for each participant. The times for participant 1 are high because we were experimenting with techniques and researching various encoding schemes. We improved the techniques further after analyzing the data for participant 2. The difficulty of the data transformation process points to the need for a specialized tool to support this process, which is currently scheduled to be developed in FY'93. This is discussed further in section 4.

Table 7. Data Transformation Times
(approximates in hours)

Participant #	Transcribe Protocol	Type in protocol split the files	Manually apply the encodings	Enter data into SHAPA
Participant 1/ Technique experimentation	42 (with keystrokes)	12	> 90	30
2	18	18 (before filter)	42	30
3	15	3.5	18	24
4	15	3.5	15	15
5 (expert)	No protocols	1.5	6	8

3.5 TRADITIONAL PERFORMANCE MEASURE RESULTS

As was mentioned earlier, it is still of interest to track the traditional performance measures. The total time used by each participant to complete the test scenario is shown below in table 8. This is followed by the completion success of each of the ten goals in the scenario, table 9.

As expected, while we learn some useful information about the system from these global measures, such as most participants had trouble successfully completing goal 5, we do not learn anything about the type of difficulty users experienced or how the system needs to be improved. There is not enough granularity in this type of data.

Table 8. Traditional Performance Measure for Each Participant

Goal	*Participant 1*	*Participant 2*	*Participant 3*	*Participant 4*	*Participant 5*
1 Set date & time	Comp	Comp	Comp	Comp	Comp
2 Review & schedule requests	Inc. (Did not schedule W zones).	Inc. (Did not schedule W zones.)	Inc. (Did not schedule 1240025 or any new missions.)	Inc. (Due to crash, chose not to re-schedule missions.)	Comp
3 Create folder FIGHTWING	Comp	Comp	Inc. (Did not press create button.)	Comp	Comp
4 Create folder BOMBTEST	Comp	Comp	Inc. (Did not press create button.)	Comp	Comp
5 Remove an airspace from a folder	Wrong (Correct procedure but removed airspace from wrong folder).	Wrong (Added airspace instead of removing. Did not press create button).	Inc. (Did not press create button.)	Inc. (Did not press create button.)	Comp
6 Enter new requests and schedule	Comp	Comp	Inc. (Folders incorrect from Goals 2, 3, and 4. Wanted to leave.)	Comp	Comp
7 Check on a request	Comp	Comp	Comp	Comp	Comp
8 Change time of a request	Comp	Comp	Comp	Comp	Comp
9 Change time of a request for a week	Comp	Comp	Comp	Comp	Comp
10 Print reports	Inc. (Forgot Canyon Run. Did not press print button for r54).	Wrong (Printed Neptune instead of Phoenix.)	Inc. (Told not to print r54.)	Inc. (Did not press print button for r54.)	Comp
Time to complete task	2.5 hrs	4.0 hrs	3.0 hrs	2.5 hrs	1.25 hrs

3.6 EXTRACTING THE USI INDICATORS AND OTHER PERFORMANCE MEASURES

From the encoded data files, measures and indicators of USI effectiveness can be extracted and summarized. We have only experimented with a few of the potential analysis techniques that could be applied to the encoded data. Our quest is to find the most useful analysis techniques and automate the application of the techniques as much as possible.

3.6.1 Measures Extractable from the Summary Data Tables

The summary data table was completed for each participant's data. This was performed manually as no tool supports the extraction of number of actions per step and number of steps per task, etc. from data. SHAPA does, however, do frequency counts so the frequency with which the predicates, e.g., task intentions and intentions to execute, occur were generated as a check that none were missed. The hierarchical encoding structure facilitated the extraction of these measures. The full data summary table for participant 2 is given in Appendix C (others can be obtained upon request), and a sample excerpt from participant 1 is included in figure 9. It is interesting to note that examining the user-interface expert's data provides us with information on the best the system can do. The users' performance provides us with information on more realistic intentions, and whether they found the system implements functions in a direct and easy-to-use fashion. If a problem can be identified from the interface expert's data it should be fixed as it will definitely affect all the users.

There is a data summary table for each goal, which corresponds to one task from the task scenarios provided to the users as part of the usability test. The first column, with heading int.task, contains a list of all the user's task intentions while performing the goal. The second column contains information on the number of times each task intention was performed. The third column contains information on the intentions to execute for each task intention (the computer step or method used to accomplish the task), followed in the fourth column by the number of times each step was performed. The fifth column contains information on the number of computer actions, at a user-interface object level, that were performed to accomplish each intention to execute. This is followed by the evaluation of the success of each of the intentions to execute in the sixth column. The column labeled evaluation of task intention (eval of int.task) provides information on the evaluation of the success of each of the task intentions. This is followed by a listing of errors and their type which occurred during the task followed by the usability analyst's comments. The comments could include information on the causes of errors, more details on errors, interesting user comments made during a task (particularly when stating desires for missing functions), the noting of trends in the data, reasons for incompletes on tasks, etc.

Interpreting the data requries familiarity with the goals, tasks, and system implementation. A person with that knowledge can see from the summary data tables the ease with which each part of the activity was completed. In figure 9 this user had the intention to set the time and date "setdate" three different times which is indicated by the 3 in the second column of the table. The first time ended in an abort due to the evaluation error; the second time ended with an incomplete

due to the failure to put the time units in EST time; and finally the task was completed correctly. Within the three occurrences of this task, the user had five intentions to execute. On at least one occurrence of the task intention, two or more executes were performed consecutively (indicated by the 5 int.execs for only 3 int.tasks). We can deduce from the variety and number of errors encountered during the intention to set the time and date that participant 1 had some minor difficulties in performing this intention. The problems included five errors; the first error (e1) was an error in evaluation which caused the user to abandon a correct sequence of steps, as she thought a different task needed to precede this one. There was also an error in intention (e5), where the user forgot to perform a part of the task; when changing the time, she neglected to put the time in EST units as specified by the task scenario. She also had one error in action specification (e3) as she had intended to change the month but omitted this action. The other errors were all minor execute errors; the user recovered from all of the errors which is indicated by the "R" following each error code.

Goal 1

Int.task	Freq	Int.exec	Freq	# actions per int.exec	Eval of int.exec	Eval of int.task	Errors	Comments
1-1 setdate	3	setdate	5	3 8, 3, 7 7	abort inc, inc, inc Ok	abort inc OK	e1 - err.eval - R e2 - typo - err.exec -R e3 - forgot to set month - err.acsp- R e4 - menutype - err.exec - R e5 - not in EST - err.int -R	Thinks must have airspaces up first
1-3 verify missions	1	move timebar	1	1	OK	OK		
1-2 set airspaces	1	set layout	1	10	OK	OK		

Figure 9. Sample from the Summary Data Tables of Participant 1 for the Goal "Set Time and Date."

On this task, the most serious problems are the evaluate error and the intention errors. Why did the user think she needed airspaces on the screen before she could set the time and date? Did the user just forget to put the units in EST time and forget to change the month field or did the nterface design contribute to this omission in some way (e.g., by having old default data filled in the fields, not making it apparent to the user that they did not alter these fields)? Could the interface have prompted the user in some way, or made the current units more obvious?

After evaluating user problems for each participant on goal 1, the next step would be to compare performance on goal 1 across users. As a baseline, we first check the user interface expert's performance. The interface expert took 7 actions to perform this task, with a frequency of one. A scan of the other users shows that three others nearly matched that performance, and only 1 other user had some errors in action specification. Based on these results, we would

recommend checking whether the interface contributed to participant 1's difficulties, but overall we would conclude the task was supported reasonably well by the interface. [Note: we later saw that this same task, when performed during other goals, did cause users difficulties and changes were recommended. This reinforces the need for realistic task scenarios to test the USI.] As the user performance for each goal is evaluated, a sheet should be kept rating each area, as well as noting USI problems to be considered for redesign.

As was mentioned previously, when assessing how many actions are too many, both frequency of the action needs to be considered as well as the task itself. If the task involves editing data in three data fields, you would expect a minimum of three actions so 4 or 5 actions would be reasonable. If the task is to change one data field and this takes 4 or 5 actions, it is cause for concern.

3.6.2 Summary Data Table Indicators for Goal 2, Scheduling Missions

Next, we will examine the users' performance for scheduling missions in goal 2. Figures 10 and 11 show excerpts from the summary tables for the expert and a user, respectively. Scheduling involved reviewing the requests, approving or disapproving them, and resolving any conflicts that migh have resulted. In order to evaluate how well the system supports the scheduling task, we need to understand the task. The task was to schedule missions, some of which consist of multiple parts (figure 6 of the MAMS display shows missions connected with lines which indicate multiple part missions). All total, there were 71 mission parts to be scheduled for goal 2.

With the MAMS prototype, there are two methods for approving requested missions. One is to select the requested mission icon with the mouse, and select the command approve from the schedule menu. The other is to select the requested mission icon and press 'control a', the command by-pass equivalent to the menu command. The first method is counted as 3 actions (select icon, select menu, select command), while the second method is counted as 2 actions (select icon, press the keys).

What indicators exist of indirectness from the user interface expert's performance? The expert averaged 2.1 actions/mission part. Two indicators of a problem would be two different levels of repetition within the task intention column and within the execute intention column. The users task is to schedule all the airspace requests. After the user has approved the requests, the system will inform the users if conflicts for airspace exist. The user then resolves the conflicts. A sensible approach to this task, therefore, is to approve all the requests for airspace, and then resolve the identified conflicts. As can be seen in column one, approving multiple mission requests is not possible. If it were, there would be a single "sch" task, preceded by selection of the missions to be scheduled. Instead, we see "sch" tasks for each mission. The user is forced to work at a lower level and form an intention to schedule each mission request individually.

Goal 2

Int.task	Freq	Int.exec	Freq	# actions per int.exec	Eval of int.exec	Eval of int.task	Errors	Com-ments
2-1 setairspaces	1	movetimebar setairspaces	1 1	1 5	OK OK	 OK		
2-2 schwpn-m	1	schwpn-m	1	2	OK	OK		
2-3 schblb-m	1	schblb-m	1	3	OK	OK		
2-4 schdact-t	3	schdact-t	3	2 2 2	OK OK OK	inc. inc OK		Not all parts of mission
2-5 schwpn-t	1	schwpn-t	1	2	OK	OK		
2-6 schwpn-w	1	schwpn-w	1	2	OK	OK		
2-7 schwpn-h	1	schwpn-h	1	2	OK	OK		
2-8 schwpn-f	1	schwpn-f	1	2	OK	OK		
2-9 schsdt-m	1	schsdt-m	1	2	OK	OK		
210 schsdt-t	1	schsdt-t	1	2	OK	OK		
2-11 schsdt-w	1	schsdt-w	1	2	OK	OK		
2-12 schsdt-h	1	schsdt-h	1	2	OK	OK		
2-13 schthawk-h	1	schthawk-h	1	2	OK	OK		
2-14 schsdt-f	1	schsdt-f	1	2	OK	OK		
2-15 schbravo-m	4	schbravo-m	4	2 2 2 2	OK OK OK OK	inc inc inc OK		not all parts of mission
2-16 schbravo-t	4	schbravo-t	4	2 2 2 2	OK OK OK OK	inc inc inc OK		

*Sch means schedule, followed by the mission name. After the dash is the day of week, with "h" used for Thursday.

Figure 10. The User Interface Expert's Summary Data Table for the Goal "Schedule Missions".

The mission show in line 2-4 is an example of a multiple part mission. As seen in column 4, the intention to execute this task was repeated three times, indicating that there was no way to apply the schedule approve command to multiple parts of a single mission. The user is again forced to work at a lower level than desired. This example differs from the first in that in the first case we were not missing a possible higher-level object; in the first case we wanted to temporarily group objects for the sole purpose of applying a command once. Here, its conceivable that the total mission with all its parts should be an object of its own. In fact, in some cases the system does consider the multiple parts as a single object, such as for dragging on the display. A conclusion we can draw so far is that the application of the scheduling commands such as approve should be reconsidered as to the level at which they can be applied.

To determine exactly how many times this Int. exec -> Tape -> Menu -> Command -> Evaluate -> Evaluate -> Int.exec was executed, we ran a frequency of predicate cycles on the expert's data. This counts the number of times the same sequence of events occurs. The results revealed this cycle occurred 62 times. Unfortunately this analysis routine, as discussed later, can only be run at the predicate level. From examining the data, however, we know that this cycle is

the approve mission request cycle of actions. From this data, we would assume that the system does not support selecting multiple missions to which a single scheduling command can then be applied; this is the case.

Figure 11 shows a real users summary data table for this same task. We first notice the same types of problems that the expert had. For multiple part missions such as r54-w which has two parts (int.task 2-32), the number of actions increases -- scheduling all of r54-w takes 4 actions implying the system does not consider the various parts of r54-w to be a single object. This results in repetitive sequences of action on the user's part.

Int.task	Freq	Int.exec	Freq	# actions per int.exec	Eval of int.exec	Eval of int.task	Errors	Comments
2-24 resconf thawk/sdt-w	1	lookthawk/sdt-w lookwpn-w movesdt-w	1 1 1	4 1 6	OK OK OK	 OK		
2-25 schsdt-w	1	schsdt-w	1	3	OK	OK		
2-26 schwpn-w	1	schwpn-w	1	2	OK	OK		
2-27 sch1240026-w	1	look1240026-w sch1240026-w	1 1	2 1	OK OK	 OK		
2-28 schgunex-w	2	lookgunex-w lookthawk schgunex-w	2 1 1	2 2 2 2	OK OK OK OK	 abort OK		 wants to schedule thawk
2-29 schthawk-w	1	movethawk-w schthawk-w	1 1	1 2	OK OK	 OK		
2-30 schfox-w	1	lookfox-w movefox-w schfox-w	1 1 1	2 1 1	OK OK OK	 OK	 e14 - conflict state - R err.inter	
2-31 resconffox/ thawk-w	1	movefox-w lookthawk-w	1 1	3 1	OK OK	 OK	 e15 - dc - R - err.exec	
2-32 schr54-w	1	lookr54-w schr54-w	1 1	2 5	OK OK	 OK		
2-33 schred-h	1	lookred-h schred-h	1 1	3 6	OK OK	 OK		
2-34 schthawk-h	1	lookthawk-h schthawk-h	1 1	6 1	OK OK	 OK	 e16 - edit under schedule - err.acsp	
2-35 schgunex-h	1	lookgunex-h schgunex-h	1 1	3 1	OK OK			

Figure 11. A User's Summary Data Tables for Goal 2 "Schedule Missions".

A new pattern is also apparent in the user's data shown in figure 11: a repetitive sequence occurs at the intention to execute level. Before approving a mission, the user peforms an execution to "look" at the mission description; this combination is represented, for example, in task 2-27 as look1240026-w, followed by sch1240026-w. A user can look at information about a mission in two ways. When a mission is selected on the display, some information about the mission appears in the documentation line at the bottom of the display. More complete

information on a mission can be obtained by selecting a mission and opening the edit dialog box. This can be performed by double clicking on the mission. This user is finding it necessary to open the edit box via a double click to obtain information about the mission before approving it; we know this because the look took 2 actions, a double click on the mission icon followed by the close button. Checking across other users' data, we see the same occurrence -- users feel a need to look at the mission information contained in the dialog box before approving the mission. This implies that these two steps should be combined in some way, to reduce the number of actions and to make it more direct. Possible solutions include identifying the key information from the edit box and putting it in the documentation line, or, if the information is varied or too long, putting an approve button in the edit dialog box (which also closes the box), so the command can be done immediately after looking, and shorten the number of steps. These indicators of indirectness are good examples that lack of user errors do not imply a direct engagement is occurring. It is also a third, distinct type of repetition from those two forms already discussed. The other examples involved the need to apply a single action to multiple objects whereas this is the need to make the display of information better and to combine action sequences. Finally, it illustrates the need for using real users in testing since the USI "expert" did not exhibit this same behavior pattern.

Some key indicators for all the users' "schedule missions" goal, including those discussed so far, are:

Indicator	*Potential Problem*
Repetitive sequences for applying the approve command to missions	Can not select groups of objects for application of a single command
Repetitive sequences for applying the approve command to parts of a single mission	System does not consider multi-part mission as an object for the case of applying scheduling commands
An abort while trying to bring up all parts of the "dact" mission on the display	System does not consider mission parts as an object for the case of finding the whole mission
An extra intention to execute required to "look" when the task intention is to schedule a mission	Information in dialog box is often required before mission can be approved. To increase feeling of directness the two steps should be combined in some manner.
Perceptual/execute errors	When the missions were physically displayed too close together, users would select the wrong one. There was no way to differentiate missions when the labels were very small.
Execute error, many actions for recovery	A user selects deny from menu rather than approve which is adjacent. Lack of undo causes user to perform multiple actions to fix.
	Mission icons were often accidentally moved. Users then had to manually reposition them. Two problems are icons are too sensitive, and there is a lack of an undo feature, resulting in multiple actions to undo a previous action.

Indicator	*Potential Problem*
An extra task intention required to "see schedule", or to find next unscheduled mission; occurred with high frequency (16, 10, 15) and many actions per step	System provides no way on main screen to allow user to automatically "jump" to: next mission, next unscheduled mission, next mission part within a mission, next conflict, etc. All movement between icons is by manually searching and manipulating.
Errors in evaluation on task completion	The system provides no indicator on the main screen of number of unscheduled missions remaining within the schedule period -- many users thought they had finished scheduling all requests when they had not. This information was available, less directly, elsewhere in the system.

3.6.3 Summary Data Table Indicators for Goal 4, Creating Folder Fightwing

As a final example of how indicators of potential problem areas can be extracted from the data, we will look at the summary tables for creating the folder named fightwing. A folder is a group or collection of airspaces which are created by the user. The folder function allows single actions to be applied to many objects at once. For example, rather than selecting individual airspaces to be displayed on the main screen, a folder can be selected for display, and all the airspaces in the folder will be displayed. This is a good function as it minimizes repetition of actions. We will soon see, however, that its implementation is not so good.

In the test scenario, users were asked to create a folder, name it fightwing, and add four airspaces to it. The first indicator of indirectness we see from the expert's summary data table (figure 12) is the high number of actions, 18, required for the intention to execute the adding of the four proper airspaces to the folder. Looking back to the articulatory-level encodings, we see that many airspaces are being selected and deleted from the folder. The way the system implements this function is to include any airspaces currently on the main display in the folder when it is first created. If some or all of these airspaces are not wanted in the folder, they need to be individually selected and deleted (itself a repetitive action as these actions can only be done on individual objects). This resulted in more actions to remove unwanted airspaces than actions required to add wanted airspaces. There are several options for increasing the directness here. One is to not have displayed airspaces default into the folder. Assuming that the system designers had a good reason for implementing the folder function this way, an alternative way to reduce the number of actions and make this more direct would be to have a function which clears the default airspaces with one action. Or, multiple selected airspaces could be deleted or added at once. A more complete redesign would involve directly dragging wanted selected airspaces into a folder icon, rather than selecting from a list and using a button.

Int.task	Freq	Int.exec	Freq	# actions per int.exec	Eval of int.exec	Eval of int.task	Errors
3-1 createfight	1	openfaf	1	2	OK		
		entername	1	2	OK		
		properairspaces	1	18	OK		
		createfolder	1	1	OK	OK	

Figure 12. Summary Data Table for User-Interface Expert on Creating Folder Fightwing

Examining the users' data will provide us with information on how direct users found the engagements required to perform this task. The summary data for the create folder fightwing goal for participants 2 and 3 is shown in figures 13 and 14. An indicator of a problem is apparent in participant 2's data. In the intention to execute column we see a "checkfight" step, which takes three actions. Also, the createfolder step itself took 2 actions, as is shown in the # of actions per int.exec column, rather than the one required, and the participant had an error in interpretation. From the comments column, we see that analyst who created the summary tables noted that after the user selected the create folder button the first time, the system provided no response. The user assumed nothing had happened as the system state did not seem to change, and pressed the button again, still trying to execute her intention. The system had indeed created a folder the first time but since it provided no indication of its change of state, the user erroneously concluded that nothing happened. The user was finally forced to perform extra actions as part of an extra step, to obtain the information necessary to correctly interpret the system state, and make the determination as to whether she had moved closer to her goal.

Participant 3 (figure 14) had even more difficulty creating a folder than participant 2. This participant was the least familiar with the system. The user had difficulty in locating the correct menu command to open the correct dialog box. The user would eventually learn this with training but looking at the menu hierarchy, we see that the correct command is nested hierarchically within a different command, making it difficult to locate via searching through the menus. Given there is plenty of space for commands, the use of hierarchical menus could probably be eliminated. Looking at the frequency count for "openfaf" we see that it takes the user three tries to open the correct dialog box within which to create folders. When the user finally found the right dialog box, and performed close to the correct sequence of actions, he then closed the dialog box without pressing the create button, and nothing was created. Given the lack of feedback to this action, the user did not notice that nothing was created. This is a case of an inexperienced user interacting with a poor design to create a very poor sequence of engagements.

Int.task	Freq	Int.exec	Freq	# actions per int.exec	Eval of int.exec	Eval of int.task	Errors	Comments
3.1 createfight	1	openfaf	1	2	OK			
		entername	1	1	OK		e.16 - typo -	Only wanted 4
		properairspace	1	17	OK		R - exec.err	airspaces. If folder came up blank could dc 4 times to complete. Have to add one by
		createfolder	1	2	OK		e.56 no created feedback err.inter	one? Remove unneeded ones also.
		checkfight	1	3	OK	OK		The create folder button gives no feedback so they have to get out, bring up folder select to see if created Not an error in interpretation but inability to interpret system state. Semantic dist on eval side. Don't know if goal was met. Check implies evaluation

Figure 13. Summary Data Table for Participant 2 on Creating Folder Fightwing.

Int.task	Freq	Int.exec	Freq	# actions per int.exec	Eval of int.exec	Eval of int.task	Errors	Comments
3-1 create fight	2	open faf	3	3	wrong		e38 - aborts file opens layout - err.eval - R	wrong DB - layout DB. Opened right one, thougt was wrong
				2	wrong		e43 - wrong DB - err.acsp -R	change layout, ok- figuring out DB
				6	OK		e50 - wrong DB err.acsp - R	Opened "create request"
		entername	2	2	wrong			
				4	OK	inc	e39 - wrong DB - err.acsp - R	inc not created, was finally getting close.
							e51 - must hit creat button after data entry - err.acsp	Finally found right DB and entered data
							e46 - changing airspaces not	and forget the create button
		proper airspaces	2	14	abort		making a folder - err.eval - R	Did he think he did it properly? Yes SN5, 308
				32	wrong	wrong		
							e40 - hits button before selecting list select - err.acsp	Realizes wrong DB
							e41, e44 - layout wrong DB - err.acsp	Change layout screen was changing as they added/ removed air-spaces and they did not notice
							e42,e45 - remaining airspace layout - err.inter - R	
3-2 check Fight	1	edit search field	1	3	OK	abort		Not there so must abort and start over

Figure 14. Summary Data Table for Participant 3 on Creating Folder Fightwing.

There are numerous examples of how indicators can be extracted from the summary data tables to provide information on indirect interactions; only a few examples were shown here. Next, we will conclude with a brief discussion of the errors broken down by stages.

3.6.4 Error Indicators

Appendix D contains all of the errors made in each stage, grouped by type, with the number of times each participant made that error. Errors are obviously useful indicators of USI problems. Each error listed in the error summary in the appendix should be examined for

potential USI improvements. That does not mean that all command names, locations, and sequences causing difficulty should be changed. Indicators of a serious problem requiring changes would be multiple participants making the same error, the same participant repeating an error, or the error has serious consequences. Comparing errors across participants also provides much information on the participants' skill level as well.

Some participants, for example, were not proficient with the mouse and made many execute errors which involved the use of the mouse. The participants also were not skilled typists and had a total of 65 typos. It is up to the system designers to decide if these problems will diminish with practice or whether the system can be modified to aid the unskilled input device user. Errors in action specification were a good indicator of the user's level of experience with the system as well as the articulatory directness. Participant 3, one of the users who was least familiar with the system, made many action specification errors, as would be expected.

Classifying Errors by Stages

Classifying errors by the stage of user activity is much more difficult than just identifying input and execution errors, as is typically done; the benefit is the greater diagnostic power of the classified errors. Classification may be difficult because the observable symptoms of errors of different types can look the same. The classification of the error involves taking into account the user's past performance at the time the error was made, his future behavior, when the error was noticed and recovered, as well as his protocols which reflect his mental processes. This analysis would be difficult to do without the combined protocol and history log files. Action specification and execute errors can look similar. If, for example, a user selects the "deny" command from the schedule menu but immediately tries to undo the results of that command and then selects "approve", which is an adjacent menu command, we would classify the error as an execution error. It was clear that the user intended to execute the approve option; the user never thought that deny was the correct action to accomplish the intention. As confirmation, there may have been a protocol stating that selecting deny was not the intention. If the user selected "deny" by accident and did not realize it right away, it would be harder to interpret whether or not this was their intended action. Looking ahead to whether he detects the error later and changes it to an approve would help confirm it was unintended.

Another case of similar symptoms with different roots would be if a dialog box was opened which did not accomplish the stated intention but whose command was adjacent to the correct one. A determination of whether the user thought the opened dialog box was correct would have to be made to know whether it was an execute error or an action specification error. If the user had opened the correct dialog box in the past to accomplish that intention, and in this instance s/he immediately corrected this problem, we would assume an execution error was made. If the user continued with a wrong sequence of actions, continued to select incorrect options, and had never performed the correct sequence of events, we would classify the error as an action specification error. An interesting case occurs when a user has correctly performed a sequence of actions in the past, and then makes an error of omission. For example, one user when changing the date/time information in the date dialog box neglected to change the duration. Since

he had correctly done so previously, we classified this as an execute error; we felt he knew and intended the correct sequence of actions but forgot to execute one action. This type of error is comparable to typing a word with an incorrect spelling. Was the error that the user did not know the correct spelling and the typing was intentional, or was the correct spelling known but the word was typed incorrectly? One is an error in spelling while one is an error in typing. If the word had been spelled correctly previously, you would suspect the manual typing error.

The classifications were performed as follows. Errors in intention, or mistakes, occurred when the user intended to do something which would not move him closer to the goal. Many of these involved forgetting or misinterpreting the task instructions. For instance, many users scheduled requests for a period of only four days, rather than five or incorrectly identified his/her agency name. The intention error served to alert us that what they were doing was not expected but we would then evaluate their performance on the remaining stages without penalty (i.e., we would not classify all their behaviors within the "wrong" tasks as errors). We would, however, evaluate the success of their endeavor with an incomplete or wrong, depending on the situation. Errors in intention, therefore, do not reveal much about the USI design, but rather about how well the users followed and interpreted the task scenario descriptions.

Errors in action specification concern the sequencing and appropriateness of user input actions for a given intention to execute, and are one of the indicators of articulatory indirectness. These included instances of the error types: presses action button before filling in the necessary data, performs button actions out of sequence, does not locate menu item on first try, does not recall/execute correct sequence of events, mixes up two dialog boxes but recognizes it is wrong one once opened, wrong concept of button/object functionality, etc. Classifying action specification errors were fairly straightforward, once the intention to execute information was known. The analyst also needs to know the correct or acceptable sequence of actions for every type of execution. Again, this analysis requires detailed information on all user inputs as well as the intention to execute associated with the inputs which is obtainable from the protocols.

There were only eight errors attributed to perceptual difficulty. These included selecting the wrong mission icon because the missions were too small with the current schedule scale setting, not noticing that a dialog box was already open and trying to reopen it, and taking actions which cause changes to the time bar which go unnoticed. Perceptual errors occur when the contributing cause of the error appeared to be imperceptible or unnoticed information.

Errors were classified as interpretation errors when the displayed information was judged to be perceivable but the user did not correctly extract its meaning, or did not correctly judge the system state. For example, if a mission turned red and the user failed to interpret this as the denied state, it was an error in interpretation. One major contributor to this error is lack of or poor system feedback which causes a wrong interpretation of the system state. Note that once an error is determined to have occurred at a lower state, we do not label as errors the following states, which may be incorrect due to the earlier error.

Finally, errors are classified as errors in evaluation when users think they have accomplished their intention and they have not, think they have not made progress toward their goal when they have, or are confused because what did happen was not what they expected. When an error was made in a different stage and not immediately noticed, we debated whether this should be considered an error in evaluation as well. We decided instead that our labeling of the success of the endeavor (incomplete or wrong if contained an uncorrected error) would reflect the unnoticed error and did not call it an error in evaluation. The success of the endeavor and the fact that the error was not immediately recovered from are themselves indicators.

The classification by stages process adds additional information by assessing at which stage the error occurred, and differentiating errors with similar observable symptoms. It involves making assessments of the users' mental activities and pinpointing the most likely stage the error occurred. This is a new concept as user input activities are all that is usually studied. Applying the classifications consistently can be difficult but it gets easier with practice.

System Design Implications Based on Error Analysis

Error frequencies are shown in table 9 below. As we have noted with the other high-level summary measures, knowing just the error total alone, e.g., 317, would not provide much diagnostic information.

Table 9. Frequency of Errors by Stage of User Activity.

	Participant #					
Errors in:	***1***	***2***	***3***	***4***	***5***	***Total***
Intention	9	10	5	5	0	29
Action specification	20	13	48	35	1	117
Execute/repositions	31	29	19	36	9	124
Perception	1	1	4	2	0	8
Interpretation	6	6	4	3	0	19
Evaluation	7	2	4	6	1	20
Totals	74	60	84	88	11	317

There were quite a few errors caused by the current USI design where improvements should be considered. For example, users had difficulty in locating the find and edit commands on the menus. There were multiple errors in action specification on the folder task. The date and the layout dialog boxes were often confused. The button labeled "change screen" appears to be ambiguous, resulting in a variety of errors. There were a large variety of execution errors, each of which should be assessed for improvements.

There were relatively few, only eight, perceptual errors. The other indicator of perceptual activity, however, the frequencies of actions taken to improve perceptibility, was quite high. There were 37 instances of participants moving the time bar to improve perceptibility of the display, and 42 instances of setting the screen to improve perception. This indicates there were many occasions when users' could not see information well on the display. Additional user aids to help improve screen perceptibility should be considered. These could include a zoom function, indicators when there are overlapping missions, adding the mission name to the documentation line and possibly a warning when the screen schedule period is set to be so long that mission perception will be impossible.

There were interpretation errors in interpreting the schedule period displayed and the time bar. Finally, for the evaluation errors, there were many cases of users thinking all the missions were scheduled when they were not. A "number of unscheduled missions" indicator would help this. As noted previously, errors in intention are not really indicative of USI design problems, but rather of human performance problems. This system supports an ill-defined creation task to create an acceptable schedule. The information displayed was not very complex and there were not a lot of information codes and very few icons. Other systems which support tasks involving interpretation of complex graphical images may result in many more errors on the evaluation side of the activity cycle than did this particular application.

In general, all errors should be evaluated and used in conjunction with the other indicators to determine if a USI change is warranted.

3.7 SUMMARY

The application of the encoding scheme to the collected usability data was very useful in assessing the directness of the engagements of the user-system interface. The data was in a form appropriate for the analysis of user engagements, allowing us to assess the interactive nature of HCI. We felt that much more information was available on the HCI process when protocols were combined with history files and encoded than if we had used any one technique alone. The combination of real users' actual task intentions with detailed information such as number of actions to perform each engagement and types of error per intention to execute allows us to measure the directness of those engagements. We were able to clearly demonstrate that traditional high-level performance measures such as task time, task completion, and error frequencies alone are inadequate for diagnosing USI improvements. With the new encoding technique, indicators of inefficient engagements are readily apparent. Patterns in the data which occur at different levels provide information on different types of system design problems. The user interface expert's data shows basic system design problems; however, the interface expert may not have realistic task goals and strategies. The actual users' data indicates how direct real users with real intentions find the system to use, as well as revealing much information on users' experience levels and individual interaction problems. Finally, the data is in a form amenable to quantitative analysis, removing much of the ambiguity which results from methods such as observation only. This was only the first step, however, and much remains to be done to improve this methodology and make it more efficient.

SECTION 4

FUTURE WORK AND CONCLUSIONS

The results obtained to date on measures of user-system interface effectiveness are very promising. We have, for the first time, a method which allows us to obtain measures on the directness of user engagements with a system. We have successfully integrated protocol data with history file data, for a complete and useful picture of HCI activity. We have created a theory-based encoding scheme which provides a method for quantitative analysis of the data. We have created an error classification scheme based on the stages of user activity model, which provides information on which stage in the human information processing cycle an error occurred and how to fix the system to prevent it than is possible to obtain from traditional error frequency measures. Indicators of USI effectiveness extending beyond errors and time have been proposed and found to be useful. We have successfully shown that a USI engagement can be error-free but not be direct, and new measures and indicators such as those proposed here are required for a complete evaluation. The measures are also in a form which allows for easy comparison across subjects. We have the ability to determine whether difficulties are due to a single user's inexperience or whether problems can be attributed to the system design.

We still have, however, many more areas to explore both in terms of the measures and indicators, and in the process for integrating the data and applying the encoding scheme.

4.1 MEASURES AND INDICATORS

We need to do several things in the area of refining the USI measures and indicators. First, we need more rigorous definitions of the different levels of the encodings; when is something a task intention as compared to an intention to execute? While we tried to be consistent in our application of these terms, it was difficult, particularly as this was the first time we applied the scheme. The same problem holds with regard to the level of detail for the intentions to execute. Sometimes all actions within a dialog box were considered to be a single intention to execute, and sometimes particular actions were broken out separately. This may need to be flexible based on the USI areas of interest.

We would like to continue to work on the definitions and names for the different USI indicators. These concepts are appealing because they provide a taxonomy that different usability specialists could use to discuss similar kinds of problems across systems. Rather than being forced to work with specific system problems, problems can be generically classified and eventually mapped to known solutions. With an ability to define a method and measures, we can specify a method for contractors to conduct usability studies.

There seem to be many kinds of repetitions which are indicators of different types and levels of problems. We would like to classify all of these various kinds of repetitions and determine what they imply for the system design.

Finally, we need to apply the encoding scheme to a different system to ensure it is generic across systems, and continue to refine it.

4.2 EXPLORING OTHER ANALYSIS ROUTINES AND THEIR EFFECTIVENESS

We need to continue investigating other analysis routines and their usefulness. SHAPA, for instance, has some built in routines for calculating frequencies, matrix analyses, lag sequential analyses, and frequency of cycles. The frequency routines were useful because the most frequently used commands could be identified. This aids in assessing how many actions are too many. For instance, the edit/view dialog box had frequencies ranging from 24 to 80 across participants. We would expect this box to be easily accessible via short cuts. It just so happens that it is; users can double click on a mission icon to bring up the corresponding edit dialog box for that mission.

The problem with the routines such as frequency of cycles was illustrated earlier. The routines only work on the predicates. The patterns, therefore, are of very high-level repetitions. For instance, Int.exec -> Menu -> Command -> Button -> Evaluate ->, is useful in that you see a dialog box is being opened and immediately closed with no actions taking place inside it, but you do not know if it was the same dialog box being opened repeatedly. On the other hand, a sequence List-select -> Button -> List-select -> Button -> List-select -> Button, would be interesting because it suggests an inability to apply a single action to many selected items in a list. If we took it to a lower level it would not show up as a pattern because each selected item in the list would be different. A more sophisticated pattern recognizer which allows wildcards is required. Identifying the different types of repetitions that could occur would allow us to select the techniques which best identify the various types. We briefly looked at another tool called the Maximal Repeating Pattern analysis tool (Siochi, 1991) but that too found repetitions only at one level.

4.3 A TOOL FOR AIDING THE APPLICATION OF THE METHODOLOGY

One apparent drawback to this method is the number of steps and time required for data transformation and integration, as well as the manual extraction of indicators and numbers of actions, steps, and task intentions. The most tedious tasks were transcribing and typing verbal protocols, reviewing the video tapes multiple times, preparing files for SHAPA, and entering data into SHAPA.

SHAPA also had many limitations which caused unnecessary work. One limitation of SHAPA was the size of data files it would accept. SHAPA designers claimed files up to 64K were acceptable but we experienced difficulty with files over 15K in size. On average, each participant's data file in our study was about 150K; therefore each data file had to be split into separate files. The splitting of data files also required manual collation of each separate file's generated reports in the analysis stage. While using SHAPA other inconveniences were encountered. Once a file has started to be encoded, the protocol file cannot be edited. SHAPA does allow splitting and combining of lines. In order to insert a line, the line must actually be split so that at least one letter is left on both lines. SHAPA does not have any copy, cut or paste feature which would be helpful. The reports also have limitations. Frequency of cycles only runs between one predicate. Tracking the cycle between two predicates would provide useful information. The value lists do not provide correlation between entities within a predicate (those separated by a coma), which is a huge drawback of this software. One other quirk of the software is that upon completion of encoding a file, the file must be closed and then reopened to run accurate reports.

Due to all of the limitations of SHAPA, which was not ever intended to support this particular method, we plan to specify requirements for a tool which will be dedicated to performing this method. The tool will be multi-media in nature and will aid in integrating the history file and the users' intentions. We may be able to remove the step of transcribing all of the users' protocols and just extracting the information needed for the intentions, evaluations, etc. Also, with a good tool, we may be able to do some encoding real-time while observing subjects. All of these areas will be looked at in the coming year.

4.4 ASSESSING THE EFFECTIVENESS OF PERCEPTUAL ACTIVITIES

Two of the stages of user activity could not be assessed as completely as we would have liked using the described methods of data collection. To understand the perceptual processing and interpretation of the display output we basically relied on errors in these stages and the frequency of input actions to aid in improving processing in these areas. We really were not directly measuring perception as could be done by identifying the number of displayed data items visually processed, or other measures of perception. Display output is becoming more graphical in nature and work to evaluate graphics and imagery to assess its effectiveness needs to be performed. This may involve different types of data collection devices such as eye-trackers.

4.5 CONCLUSIONS

This research has provided us with a much better understanding of what is required to do a complete user-system interface evaluation. We now have a theory-based framework within which concepts such as semantic and articulatory distance can be measured, and we have created operational, working definitions and indicators for these concepts. We were successful in integrating protocol and history data for a complete description of HCI activity. We created a generic encoding scheme for abstraction of data. We validated the methodology by applying it to a real prototype. We are now one step closer to being able to measure the usability of a system in a quantifiable terms.

Much remains to be done however. The current process for applying this methodology is time-consuming and tools need to be developed to support the process. We need to apply the method to another system to confirm the measures and we need to continue to refine the indicators as well. These are the goals for this project as it continues through FY93.

SECTION 5

REFERENCES

Card, S. K., Moran, T. P., & Newell, A. (1983). *The Psychology of Human-Computer Interaction.* Hillsdale, N. J.: Lawrence Erlbaum Associates, Inc.

Holleran, P. A. (1991). A Methodological Note on the Pitfalls in Usability Testing. *Behaviour & Information Technology*, 10, 5, 345-357.

Hutchins, E. L., Hollan, J. D., and Norman, D. A. (1986). Direct Manipulation Interfaces. In D. A. Norman and S. W. Draper (eds.) *User Centered System Design: New Perspectives on Human-Computer Interaction.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Landauer, T. K. (1987). Relations between Cognitive Psychology and Computer System Design. In J. Carroll (Ed) *Interfacing Thought: Cognitive Aspects of Human-Computer Interaction.* Cambridge, MA: The MIT Press.

Lewis, C., Polson, P., Wharton, C., & Rieman, J. (1990). Testing a Walkthrough Methodology for Theory-Based Design of Walk-Up-and-Use Interfaces, *CHI '90 Proceedings*, 235-242.

Norman, D. A. (1986). Cognitive Engineering. In D. A. Norman and S. W. Draper (eds.) *User Centered System Design: New Perspectives on Human-Computer Interaction.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Prasse, M. J. (1990). The Video Analysis Method: An Integrated Approach to Usability Assessment. *Proceedings of the Human Factors Society 34th Annual Meeting*, 400-404.

Sanderson, P. M., James, J. M., and Seidler, K. S. (1989), SHAPA: An Interactive Software Environment for Protocol Analysis. *Ergonomics*, 32 (11), 1271-1302.

Siochi, A. C. and Hix, D. (1991). A Study of Computer-Supported User Interface Evaluation Using Maximal Repeating Pattern Analysis, *Proceedings of CHI'91*, 301-305.

Tullis, T. S. (1983). The Formatting of Alphanumeric Displays: A Review and Analysis. *Human Factors*, 25(6), 657-682.

Virzi, R. A. (1992). Refining the Test Phase of Usability Evaluation: How Many Subjects is Enough. *Human Factors*, 34, 457-468.

Whiteside, J., Bennett, J., Holtzblatt, D. (1988). Usability Engineering: Our Experience and Evolution. In M. Helander (ed.) *Handbook of Human-Computer Interaction*, N. Holland, Amsterdam: Elsevier Science Publishers.

Ziegler, J. E. and Fahnrich, K. P. (1988). Direct Manipulation. In M. Helander (ed) *Handbook of Human-Computer Interaction* , N. Holland, Amsterdam: Elsevier Science Publishers.

APPENDIX A

TASK SCENARIO

For the following scenario you will be acting as a scheduler for the Phoenix Agency. The Phoenix Agency has a number of Special User Airspaces (SUAs) for which you will be responsible. These SUAs are: Canyon Run, Yankee 1, Yankee 2, India, W-556A, W556B, W556C, R-7221, R-7222 and R-7223 which is subdivided into R-7223N, R-7223S, R-7223E, and R-7223W. All of your airspaces are active or available for missions to be scheduled into them Monday through Friday from 0600 EST (1100 Z) to 1800 EST (2300 Z) except for India which is available 24 hours per day.

You have access to viewing and requesting SUAs in other agencies but you do not have authorization to schedule those airspaces.

1) You are planning a schedule for the week of 13-17 April 1992. All of the work done at Phoenix agency is done on EST. Set the screen start date and time appropriately.

2) Look at the requests for the airspaces you control, deny, or edit them as you deem appropriate. You cannot accept any conflicts.

3) Since you will be entering a number of missions that involve the same airspaces, create a folder named FIGHTWING that contains the following airspaces: Canyon Run, Yankee 1, Yankee 2, and India.

4) Create another folder named BOMBTEST that contains the following airspaces: R-7221, R-7222, and all the airspaces in R-7223.

5) Remove India from folder NIGHTRUN.

6) The attached requests have arrived by fax. Input them into the MAMS system as approved missions. If possible resolve any conflicts. You may do this by changing the start time of a mission, denying the mission, or changing the airspace if necessary. You may not accept any conflicts.

7) A squadron that does not have access to the MAMS system has asked you to check on their request called ASR on the 13 April 92 for W-555 in Neptune NAS. Has the request been scheduled, looked at (or not looked at), or denied? They also want to know about missions with the following MAMS numbers: 1230000 in R-8722W and 1280000 in W-554. Write the status of the mission on the back of this paper and set the paper aside to be faxed to the squadron.

8) Since ASR has been denied, the squadron has asked you to change the time of the request to 13 April 92 1300 EST.

9) You have been asked to change Bravo77 to a start time of 0900Z. Bravo77 has been scheduled daily over the next week in R-7223.

10) Print the following reports:
- All missions for R-7222 and Canyon Run for the week of 13-17 April 1992.
- All missions requested by Phoenix for the week of 13 April 1992.
- Print Raider54 scheduled for 17 April 1992.

APPENDIX B

SEMANTIC AND ARTICULATORY LEVEL ENCODINGS

Semantic Encodings

GOAL

GOAL (Goal # - Name)

Goal # : Corresponds to scenario number
Name : Brief description

Possible encodings :

1 - setdate
2 - schedulemissions
3 - fightwing
4 - bombtest
5 - nightrun
6 - newmissions
7 - status
8 - timeasr
9 - timebravo
10 - print

INTENTION OF TASK

INT.TASK (Goal # - Task # - Name)

Goal # : 1 - 10
Task # : 1 - ∞
Name : Subject to the users understanding of the software

INTENTION TO EXECUTE

INT.EXEC (Goal # - Task # . Exec # - Name)

Goal # : 1 - 10
Task # : 1 - ∞
Exec # : 1 - ∞
Name : Subject to the users understanding of the software

INTENTION EXTRA

INT.EXT (Goal # - Ext # - Name)

Goal # : 1 - 10
Ext # : 1 - ∞
Name : Subject to the users understanding of the software

INTENTION PERCEPTION

INT.PERCEPT (Goal # - Percept # - Name)

Goal # : 1 - 10
Percept # : 1 - ∞
Name : Subject to the users understanding of the software

ERROR IN INTENTION

ERR.INT (Error # * Goal # - Task # . <Exec #> - Name - Problem)

Error # : 1 - ∞
Goal # : 1 - 10
Task # : 1 - ∞
Exec # : 1 - ∞
Name : Subject to the users understanding of the software
Problem : What the error is

ERROR IN ACTION SPECIFICATION

ERR.ACSP (Error # * Goal # - Task # . Exec # - Name - Problem)

Error # : 1 - ∞
Goal # : 1 - 10
Task # : 1 - ∞
Exec # : 1 - ∞
Name : Subject to the users understanding of the software
Problem : What the error is

ERROR IN EXECUTE

ERR.EXEC (Error # * Goal # - Task # . Exec # - Name - Problem)

Error # : 1 - ∞
Goal # : 1 - 10
Task # : 1 - ∞
Exec # : 1 - ∞
Name : Subject to the users understanding of the software
Problem : What the error is

ERROR IN PERCEPTION

ERR.PER (Error # * Goal # - <Task #> . <Exec #> - Name - Problem)

Error # : 1 - ∞
Goal # : 1 - 10
Task # : 1 - ∞
Exec # : 1 - ∞
Name : Subject to the users understanding of the software
Problem : What the error is

ERROR IN INTERPRETATION

ERR.INTER (Error # * Goal # - <Task #> . <Exec #> - Name - Problem)

Error # : 1 - ∞
Goal # : 1 - 10
Task # : 1 - ∞
Exec # : 1 - ∞
Name : Subject to the users understanding of the software
Problem : What the error is

ERROR IN EVALUATION

ERR.EVAL (Error # * Goal # - <Task #> . <Exec #> - Name - Problem)

Error # : 1 - ∞
Goal # : 1 - 10
Task # : 1 - ∞
Exec # : 1 - ∞
Name : Subject to the users understanding of the software
Problem : What the error is

EVALUATION

EVALUATE (Goal # - <Task #> . <Exec #> - Name - State)

Goal # : Corresponds to the goal being evaluated
Task # : Corresponds to the task being evaluated
Exec # : Corresponds to the execute being evaluated
Name : Corresponds to the name being evaluated
State : Abort - Abandons corresponding goal, task, or execute
Inc - Has not fully completed corresponding goal, task, or execute
Ok - Proper completion corresponding goal, task, or execute
Wrong - Has completed corresponding goal, task, or execute incorrectly

RECOVERY OF AN ERROR

REC.ERR (Error #)

Error # : 1 - ∞

Anytime an error has been acknowledged and recovered, it is noted as a recovery of an error. If the experimentor verbally helped with the recovery of an error it is noted as REC.ERR(Error # - help). If the experimentor actually pressed the keys to help the user get out of a bind, it is noted as REC.ERR(chip typing start) and REC.ERR(chip typing stop) and the user does not get credit for recovery.

MISCELLANEOUS

Q (Miscellaneous)

Anything out of the ordinary. For example, a crash would be noted as Q(CRASH).

Articulatory Encodings

MENU

MENU (Name, Function)

Name : Menu name
Function : dc = double click
k = key
m = mouse

Possible Menu Names :

File
Folder-hier
Mission
Schedule
View

COMMAND

COMMAND (Name, Function)

Name : Command name
Function : dc = double click
k = key
m = mouse

Possible command names within each menu name:

File (admin, admin-fold, print)
Mission (create, edit, find)
Schedule (approve, deny, describeconf, pendreq, unschedule,)
View (date, layout)

Abbreviations : Admin - administrative, Describeconf - describe conflicts, fold - folder, Pendreq - pending requests,

LIST SELECTION

LIST_SELECT (Name)

Name : Name of selection box

Possible list select names:

Create
Describeconf
Edit
Find
Folder - 1

Folder - r
Layout - undis
Layout - dis
Print
Pendreq

Abbreviations : Folder - l - available suas (left box), Folder - r - folder suas (right box), Layout - undis - undisplayed suas, Layout - dis - displayed suas

FIELDS

FIELD (Dialog Box - Name - <Function>)

Dialog Box : Same as command name except admin-fold is folder
Name : Field name
Function : data - entered into a blank field
del - deleted information in field
edit - entered into already occupied field or an empty field and editing took place while typing
<No function> - field selected

Possible field names within each dialog box:

Date (date, time, durdays, durhrs)
Create, Edit (name, type, prior, ord, unit, call, #air, airtype, sua, stdate, sttime, spdate, sptime, dur, lowalt, upalt, poc, phone, comment, label)
Find (stdate, sttime,spdate, sptime, sua, reqagency, mams#, name)
Folder (typein, search)
Layout (search)
Pendreq (stdate, sttime,spdate, sptime, sua, reqagency)
Print (stdate, sttime,spdate, sptime, name, sua, reqagency, util)

Abbreviations : alt - altitude, call - callsign, comment - remarks, dur - duration, durhrs - duration in hours, durdays - duration in days, name - mission name, poc - person on call, prior - priority, ord - ordnance, time stdate - startdate, sttime - starttime, spdate - stopdate, sptime - stoptime, type - mission type, typein - creating or finding a mission field, util - utilization, #air - # aircrafts

BUTTON

BUTTON (Dialog Box - Name - <Function>)

Dialog Box : Same as command name except admin-fold is folder
Name : Button name
Function : dc - double click
<No Function> - button selected

Possible button names within each dialog box :

Create (create, createcon-ok, createcon-cancel, cancel, ins, del)
Date (ok, cancel)
Describeconf (cancel)
Edit (edit, editcon-ok, editcon-cancel, cancel, ins, del)
Find (find, view, change, cancel, reqonly, approveonly, both)
Folder (add, rem, openfolder, create, close)
Layout (add, rem, openfolder, ok)
Pendreq (find, view, change, cancel)
Print (view, print, cancel)

Abbreviations : con - confirmation box, del - delete, ins - insert, rem - remove, reqonly - request only

SCROLL BARS

SCROLL (List select in Dialog Box)
Dialog Box : Same as command name except admin-fold is folder

Possible scrolls:

Create
Describeconf
Edit
Find
Folder
Layout
Pendreq
Print

FORMS

FORM (Name)
Name : Form name

Anytime a user hits the background ot the interface, either by mistake, missing a button or menu or misunderstanding some functional capability.

Possible form names :

Create
Date
Datebox
Describeconf
Edit

Find
Folder
Layout
Main
Pendreq
Print

TIMEBAR

TIMEBAR (Duration/Code)

Duration : The time between keypress and key release. The duration may be due to users decision making or system reponse time.

Code : p - manipulation of timebar related to perception
s - manipulation related to scheduling

MISSIONS

TAPE (Name - Day of week - Time/Code)

Name : Mission name

Day of week : Original requested date of mission (m, t, w, h, f)

Time : The time between key press and key release

Code : p - moving a mission to improve perception
s - moving a mission to schedule
r - reposition of mission due to mission slippage

MISCELLANEOUS

Q (Miscellaneous)

Anything that does not apply to the above criteria. An example is pressing a key and no computer response, i.e., Q(no computer response). Another is a crash, noted as Q(crash-recovery-start) and Q(crash-recovery-end-<evalutaion>).

APPENDIX C

SUMMARY DATA TABLES

PARTICIPANT 2
Goal 1

Int.task	Freq	Int.exec	Freq	# actions per int.exec	Eval of int.exec	Eval of int.task	Errors	Comments
1-1 Settime	1	Settime	1	7	OK	OK	e1 - 4 days scheduled - err.int - R e50 - trouble selecting field - err.exec - R e51 - trouble selecting field - err.exec - R e52 - trouble selecting field - err.exec - R	Trouble selecting all of field to delete (took 6,3,6 clicks) Hilite first data entry field in D6 - VC She then discovered the tab key and did much better Line 68 - set duration of schedule for 5 days but needs to change default setting of time for lenght

Goal 2

Int.task	Freq	Int.exec	Freq	# actions per int.exec	Eval of int.exec	Eval of int.task	Errors	Comments
2.1 Set Airspaces	1	setairspaces	1	18	OK	inc	e2 - removed w-airsp - err.int	miss w-airspaces wanted to look at all parts of Bravo 77 at once (265-269) Wants to approve all parts of mission at once (line 321) wanted to hilite more than 1 airspace to be removed from list

2.2 schwpn-m	1	lookwpn-m schwpn-m	1 1	2 3	OK OK	OK		If the info they are looking at in edit box could be displayed ??? would reduce many steps
2.3 schsdt-m	1	looksdt-m	1	3	OK			
		schsdt-m	1	3	OK	OK		
2.4 schbravo-m	1	lookbravo-m	1	2	OK			
		schbravo-m	1	8	OK	OK		
2.5 schfox-m	1	lookfox-m schfox-m repositfox-m	1 1 1	3 4 4	OK OK OK	 OK	e3 - timedday - err.eval	Count her ??? exec's as exec. errors Moved tape by accident. Exec.err? Wants a reset or undo key. Would have potentially reduced steps here
2.6 schr54-m	1	lookr54-m schr54-m	1 1	3 4	OK OK	OK		Wants to schedule missions all at the same time
2.7 sch1240024-m	1	look1240024-m sch1240024-m	1 1	3 1	OK OK	OK		

2.8 see schedule	6	move timebar	3	4	OK	OK		
				2	abort		e11 - snapped to bar - err.exec	Need to hold in final position for a second or snaps back
				1	OK	OK		
		pending	4	1	abort	abort	e6 - looks under sched.	
				1	OK			Change mind, wants to do something else
				2	OK		menu - err.acsp	
		setlayout	1	2	OK		e12 - layout/date DB-R	Wanted to open time/date db, chose set layout instead. Layout is a confusing term and should be changed
		settime	1	4	OK	abort		
		mission to screen	1	5	OK	OK	e15 - change/view button - err.acsp	Need improve perception Found mission in find d/box and wants to put upon screen. Hit "view" button - thought would put on screen - should change to read or describe
2.9 schblb-m	1	lookblb-m	1	2	OK			
		schblb-m	1	1	OK	OK		
2.10 schthawk-m	1	lookthawk-m	1	2	OK			
		schthawk-m	1	1	OK	OK		
2.11 sch1080000-m	1	look1080000-m	1	2	OK			
		sch1080000-m	1	1	0K			
		reposit1080000-m	1	3	OK	OK		Exec error. Tape moves when she selects it and it takes several moves to reposit. No undo
2.12 schwpn-t	1	lookwpn-t	1	2	OK			
		schwpn-t	1	1	OK	OK		

2.13 schdoct-t	1	lookdoct movedoct-t schdoct-t	1 1 1	4 2 6	OK OK OK	 OK	e4 - close Mssions-R - err.per e5 - dc-R - err.exec	Missions too close and selected wrong one She inadvertenly dc while selecting. This user is not good with the mouse.
2.14 schsdt-t	1	looksdt-t lookthawk-t movesdt-t schsdt-t	1 1 1 1	5 5 1 1	OK OK OK OK	 OK	e53 - trouble dc-R - err.exec e54 - trouble dc-R - err.exec	Seems to be moving it also
2.15 schthawk-t	2	lookthawk-t schthawk-t	2 2	3 1 1 16	OK OK OK	OK OK	e7 - deny rather than approve-err.exec - R e8 - time delay - err.eval	Deny rather than approve. Took several selections to correct error. Undo would have helped again. Need to unschedule and then approve 2 thawks on Tuesday
2.16 schbravo-t	1	lookbravo-t schbravo-t	1 1	2 8	OK OK	 OK		
2.17 schred-t	1	lookred-t schred-t	1 1	2 5	OK OK	 OK	e9 - dc - err.exec. - R	Accidental double click Note - numerous mouse problems. Is mouse too sensitive?

2.18 sch1240025-t	1	look1240025-t checkname change-t edit1240025 name-t sch1240025-t	1 1 1 1	3 1 4 1	OK OK OK OK	 OK	e10 - must use edit button to save changes - err.acsp - R	changing label on screen from MAMS# to Mission name. Made change but "cancelled" out of dbox Changed field, pressed "change" button, confirmed, and still had to press "cancel" button - misleading action seq. Should be "close"
2.19 schwpn-w	1	lookwpn-w schwpn-w	1 1	2 1	OK OK	 OK		Wants documentation line to show ??? and remarks pertaining to A/C operation.
2.20 schsdt-w	1	looksdt-w schsdt-w	1 1	1 2	OK OK	 	 OK	

Int.perc	Freq	Int.exec	Freq	# actions per int.exec	Eval of int.exec	Eval of int.perc	Errors	Comments
2.24 setscreen	4	movemission movetimebar	3	2 4 3 2	OK OK OK OK	OK OK OK abort		 Said "enough messing w/this (timebar), and went to pending request instead to see what was going on Wanted to "unstack" mission icons that are on top of each other

Int.task	Freq	Int.exec	Freq	# actions per int.exec	Eval of int.exec	Eval of int.task	Errors	Comments
2.21 schbravo-w	1	lookbravo-w repositbravo-w schbravo-w	2 1 1	1 2 2 8	abort OK OK OK	 OK		Tried to dc on tape; moved accidentally. Had to stop to reposition. Need undo.
2.22 schfox-w	1	lookfox-w schfox-w	1 1	2 1	OK OK	 OK		Wants to be able to select missions with the keyboard. Is this possible.
2.23 schthawk-w	1	lookthawk-w schthawk-w	1 1	2 1	OK OK	 OK		
2.25 schgunex-w	1	lookgunex-w movegunex-w schgunex-w	2 1 1	3 3 1 2	OK OK OK OK	 OK		
2.26 schr54	1	lookr54-w schr54-w	1 1	2 3	OK OK	 OK		Try to eliminate the "looks"
2.27 axg1240026-w	1	look1240026-w sch1240026-w	1 1	2 1	OK OK	 OK		
2.28 schwpn-h	1	lookwpn-h schwpn-h	1 1	2 1	OK OK	 OK		
2.29 schodt-h	1	looksdt-h movesdt-h schsdt-h	1 1 1	2 2 1	OK OK OK	 OK		
2.30 schthawk-h	1	schthawk-h	1	2	OK			
2.31 schbravo-h	1	lookbravo-h schbravo-h	1 1	2 8	OK OK	 OK	e.14 - dc - R exec.err	
2.32 schthawk-h	1	lookthawk-h schthawk-h	1 1	2 1	OK OK	 OK		
2.33 schred-h	1	lookred-h schred	1 1	4 3	OK OK			
2.34 schgunex-h	1	lookgunex-h schgunex-h	1 1	2 1	OK OK			

2.35 schr54-f	1	lookr54-f schr54-f	1 1	2 3	OK OK	OK		Wants ability to tab to next piece of a connected mission. Eliminate the visual search and potential screen repositioning
2.36 schwpn-f	1	lookwpn-f schwpn-f	1 1	2 1	OK OK	OK		
2.37 schsdt-f	1	looksdt-f schsdt-f	1 1	5 1	OK OK	OK		
2.38 schbravo-f	1	lookbravo-f	1 1	2 9	OK OK	OK		
2.39 schthawk-f	1	lookthawk-f schthawk-f	1 1	4 1	OK OK	OK	e.55 trouble dc - R - err.exec	trouble dbt. clicking
2.40 schfox-f	1	lookfox-f schfox-f	1 1	2 1	OK OK	OK		
2.41 sch1240028-f	1	look1240028-f sch1240028-f	1 1	1 7	OK OK	OK		time delay error?
2.42 sch1240027-f	1	look1240027-f sch1240027-f	1 1	2 1	OK OK	OK		
2.43 resconf 1240027/1240028-f	1	look1240027-f move1240027-f	1 1	2 1	OK OK	OK		

Int.task	Freq	Int.exec	Freq	# actions per int.exec	Eval of int.exec	Eval of int.task	Errors	Comments
3.1 createfight	1	openfaf	1	2	OK			
		entername	1	1	OK		e.16 - typo - R -	
		properairspace	1	17	OK		exec.err	only wanted 4 airspaces. If folder came up blank could dc 4 times to complete. Have to add one by one?
		createfolder	1	2	OK		e.56 no create feedback err.inter	Remove unneeded ones also. The create folder button gives no feedback so they have to get out,
		checkfight	1	3	OK	OK		bring up folder select to see if created Not an error in interpretation but inability to interpret system state. Semantic dist on eval side. Don't know if goal was met. Check implies evaluation

Int.task	Freq	Int.exec	Freq	# actions per int.exec	Eval of int.exec	Eval of int.task	Errors	Comments
4.1 createbrmb	2	entername	1	2	OK			
		properairspace	1	18	inc	inc	e.51 added wrong airspace - err.exec. - R	why? she didn't have the proper airspace in the folder
		createfolder	1	1	OK	OK		

Int.task	Freq	Int.exec	Freq	# actions per int.exec	Eval of int.exec	Eval of int.task	Errors	Comments
5.1 remove India	2	openfight	1	1	OK	abort	e.18 opened folder left - err.acsp	explain: on left, can see existing folders & can add new folders to right hand side from there, but can't edit or create new folders on left - very confusing never checked or com-pleted correctly
		findfight	1	3	OK		e.19 opened folder left - err..acsp	
		openfight	1	1	wrong		e.20-edit left - err.acsp	
		remindia	1	3	wrong	wrong		

Int.task	Freq	Int.exec	Freq	# actions per int.exec	Eval of int.exec	Eval of int.task	Errors	Comments
6.1 createthunder-m	1	open form	1	2	OK			Data entry is step-intensive
		enterinfothunder-m	1	15	OK		e.21, e.22, e.23, e.24, e.25 - typo - R - err.exec	
		createthunder-m	1	4	OK	OK		
6.2 createthunder-w	1	enterinfothunder-w	1	3	OK			
		createthunder-w	1	3	OK	OK		
6.3 createthunder-f	1	enterinfothunder-f	1	3	OK			
		createthunder-f	1	3	OK	OK		
6.4 createpara	1	enterinfopara	1	19	OK			
		createpara	1	3	OK			
6.5 create appro	1	enterinfoappro	1	33	OK		e.26, e.27, e.28, e.29 - typo - err.exec. - R	why so high?
		createappro	1	3	OK	OK		
6.6 setairspaces	1	setlayout	1	15	OK	inc		no w-zones still

6.7 createcon	1	open forms enterinfocon createcon	1 1 1	3 18 3	OK OK OK	 OK	e.30, e.31, e.32, e.33 - typo - - err. exec e.49 - ??? label & changed name - didn't realize it - err.acsp.	why did it happen? Is there 2 areas called remarks?
6.8 createfly	1	enterinfofly createfly	1 1	28 4	OK OK	 OK		
6.9 findconflicts	2	pendreq find	2	1 7 22 10	OK OK OK OK	 OK OK	e.59 - wants find not pendreq - R - err.acsp e.35 - narrow conflicts to Neptune requests - err.int. e.38 - week not 14-18th -err.int. e.39 - narrowing to Neptune - err.int. e.34 ??? - R - err.acsp e.57 - problem evaluat-ing error - no feedback - err.inter.	Intention is to find conflicts - "wish there was a conflict search" Searched using find mission form by scrolling through status of approved missions - not very direct Search found nothing due to case sensitivity error but provided no FB She thought she was "Neptune". An artifact of the experimental nature of the study.
6.10 resconf	1	changescreen lookcon-t lookthawk-t movecon-t	1 1 1 1	3 2 1 6	OK OK OK OK	 OK		
6.11 resconfdact/cln-t	1	lookdact-t lookcon-t movethawk-t movered-t movecon-t movedact-t	1 1 1 1 1 1	2 3 1 1 3 3	OK OK OK OK OK OK	 OK	e.36 - too early - err.int.	

6.12 resconf con/cf6615-t	1	describeconf	3	4 3 3	OK OK OK			After getting info on the mission in conflict, she wanted to find the missions in conflict. Had to do a separate set of steps. No find option on describconf dial box Had to open 3 times to check mission name in between, trying the various methods to locate the mission
		searchcf6615-t	2	4 2	abort abort			
		findcf6615-t	2	5 9	abort OK		e.37 - remembered wrong name - err.int - R	Could not locate the mission she was in con-flict with on main screen
		movecf6615-5	1	2	OK		e.38 - no feedback - err.int.	No FB on unsuccessful search

Int.task	Freq	Int.exec	Freq	# actions per int.exec	Eval of int.exec	Eval of int.task	Errors	Comments
7.1 statusasr	1	findasr	1	22	OK	OK	e.41 - case - err.acsp e.40 date wrong - err.int. - R	Had old date filled in. Need a "clear" button to remove already filled in date field in dboxes
7.2 status1236000	1	find1230000	1	8	OK	OK		
7.3 status1280000	1	find1280000	1	2	OK	OK		

Int.task	Freq	Int.exec	Freq	# actions per int.exec	Eval of int.exec	Eval of int.task	Errors	Comments
8.1 changeasr	1	findasr editasr	1 1	8 2	OK OK	 OK		
2.1 findconflicts	1	find	1	18	abort	OK	e.42 - narrowed to Neptune - err.int.	See missions on screen so doesn't need findform
2.2 resconflicts-m	1	lookthunder-m movethunder-m remove thunder-m	1 1 1	5 5 40	OK OK crash	 crash		has to unschedule each piece of a mission individually, also

Int.task	Freq	Int.exec	Freq	# actions per int.exec	Eval of int.exec	Eval of int.task	Errors	Comments
9.1 changebravos	1	findbravos	1	12	OK	OK		
9.3 changebravo-m	1	movebravo-m	1	2	OK	OK		
9.4 changebravo-t	1	movebravo-t	1	1	OK	OK		
9.5 see schedule	2	movetimebar	2	1 1	OK OK	OK OK		
9.6 changebravo-w	1	movebravo-w	1	4	OK	OK		moving on main screen - trying to fine tune position? Yes, trying to get exactly on 9:00. System should "snap to" nearest time unit speci-fied.
9.7 changebravo-h	1	movebravo-h	1	1	OK	OK		
9.8 changebravo-f	1	movebravo-f	1	3 (2)	OK	OK		

Int.perc	Freq	Int.exec	Freq	# actions per int.exec	Eval of int.exec	Eval of int.perc	Errors	Comments
9.2 setscreen	4	movetimebar movemission	2 2	3 (1) 4 (3) 4 2	OK OK OK OK	OK OK OK OK		
9.10 line	1	touchline	1	15	OK	OK		What doing? Big old line on screen

Int.task	Freq	Int.exec	Freq	# actions per int.exec	Eval of int.exec	Eval of int.task	Errors	Comments
9.9 checkbravos	1	findbravos	1	8	OK	OK		
10.1 getform	1	openform	1	2	OK	OK		
10.2 printr7222	1	infor7222	1	16	OK	OK	e.43 - not narrowed for week - err.int - R e.44 - typo - R - err.exec	Viewed report & realized didn't narrow view to week of interest lack of feedback for proper evaluation of success of print
10.3 print crun	1	infocrun	1	3	OK	OK		
10.4 print phoenix	1	infoneptune	1	11	OK	wrong	e.46 - printed Neptune not Phoenix - err.int. e.45 - and/or functionality - err. acsp	
10.5 print r54	1	infor54	1	20	OK	OK	e.47 - and/or functionality - err.acsp e.48 - print before view - R - err.acsp	

APPENDIX D

Error Summary

This appendix contains information on errors user's made during the usability testing, classified by stage. Each user was assigned a unique letter code, and each error an error number.

Errors in Intention

Type	Instance	Subj/error #	Freq
Misinterprets task description	4 days scheduled	T-1, N-5,34	3
	Remove W areas	T-2	1
	Prints Neptune	T-46	1
	Prints r7222 for 2 wks	T-43	1
	Reading wrong part of task	L-3,4,5	3
	Not in EST	K-5, N-3	2
	Wrong folder	K-27	1
Dialog box purpose	Folder and layout	L-37	1
Memory	Wrong mission name	T-37	1
	Wrong scheduling week (14-18 Apr)	T-38, K-57	2
	Thinks agency is Neptune	T-35, 39, 42; K-59	4
	Narrows search	N-7, K-8,11	3
	Wrong dates searching for 1280000	N-63	1
	Moved too early/wrong time frame	T-36, K-13,72	
	Thought entered wrong times	K-38	
	Thinks Phoenix is an SUA	L-79	

Errors in Action Specification

Type	Instance	Subj/error #	Freq
Presses action button before filling in necessary data	Creating para, hits button before entering data.	N-81	1
	Doesn't enter info before find button	N16	1
	Hits button open folder before entering info	N-55	1
	Select view without info. in field	N-8	1
Forgets a button in correct button sequence	In pendreq, tries to view before find	N-62,64	2
	In find, hits view before find	N-75	1
	Forgets edit button and closes	T-10,L-15,17, 19,21, N-19,21	7
	Prints before viewing in Print	T-48, L-78	2
	Forgets create button	L-7, N-51, 54, 56	4
	Needs to reselect find button to reflect changes	L-27	1
Can't locate menu item (command)	Looks for pendreq		
	- under schedule	T-6	1
	- under file admin	N-6;	1
	Find	L-8,25,74, K58,N15;	5
	Edit	L-13,65 N-18,20,30,78, K58;	7
	Layout	L-36	1
	Approve	L53, N12	2
	Admin-folder	L-77, N47	2
	Date	N-1,2,35	3
	Create	N-70	1
	Deny	K55	1

Can not recall/execute correct sequence of events to accomplish intention	In create, need ASR selected to find MAMS#	N-71	1
	Forgets how to identify mission in conflict with.	L-73	1
	Changing layout instead of creating folder.	N-41,44,49	3
	Doesn't know how to get rid of blank screen	N-59	1
	Opened folders on the left side to create	T-18,19, K-28	3
	Opened folders on left to remove an airspace	T-20, K-30	2
	Can not find location of requested missions	N10	1
	Thinks move through schedule with a menu item (vs. timebar)	K25	1
	Does not remember how to check on ASR	N67	1
Mixes up dialog boxes (realizes when it's open its the wrong one)	Date vs layout	T-12,L-48,N-4;	3
	Layout/find	K-24	1
	Pendreq rather than find	T-59	1
	Create rather than edit	L-14, N-57,73	3
	Change layout for folders	N-39,43,48	3
	Create req. rather than folders	N50	1
	Print rather than check ASR one?	N60	1
	Layout vs view	K-49	1

Wrong concept of button/object functionality	Change screen vs. view	T-15	1
	Uses time bar to try to get missions on schedule	N-11	1
	Thought cancel canceled mission rather the Dbox	K-39	1
	Thought change screen clears screen	K-51	1
	Thought change layout improves perception	N-17	1
	In find db, hits change screen rather than find button	K10,12	2
Types in letter of unacceptable case	Six instances in Find Dbox	T-34, 41, K-67, 62, 7, L-10	6
	In Print preview	L-81, K-64	2
	In create	K-45	1
And/or functionality confusion	Print Neptune	T-45	1
	Print r54	T-47	1
	Settime-and/or dur for days and hrs. Not clear if need to fill in both.	K-71	1
Tried dragging an item vs using button for action	Proper airspaces	L-2	1
Wants to do but can't from current location	Wants to edit while in the view form	T-9, 12	2

Correct object not selected as recipient of command/changes	Wants to edit Brave-T but still M	L-16,28	2
	Wants W, still M	L-18	1
	Wants H, still M	L-20	1
	Item viewed is not same as item selected for editing	L-30	1
	Add button selected before any SUAs in list selected	L-38	1
	Open folder button selected before any folders in list selected	L-39	1
	Approves without selecting mission	L-54,55, N-13	3
	Airspaces - button before list select	N40	1
Can not specify correctly due to previous error	Enter info thunder - no folder so SUA field empties out	L-35	1
Changing info in wrong field for intention	Changed total duration, not day duration	L-49	1
	Changed stop, not start	K-54	1
Enters incorrect info. into field	Enters a 6 vs 7 digit #	N-72	1
	Wrong alt. format	C-6	1
Looks for info in wrong place	Info was in doc. line, but looked in edit dbox	L-51	1
Forgets how to use hierarchical menus		N-53	1
Incorrect command by-pass key	Ctrl S for approval	K-6	1

Errors in Execution

Type	Instance	Subj/error #	Freq
Does not delete all info. in a field		N-65,68,69	3
Entered data in wrong field		L-23, C-5	2

Wrong mission		C-2	1
Trouble double clicking		T-53,54,58	3
Trouble deleting field info		T-50,51	2
Added wrong airspace		T-17	1
Typo		totals: K-24 T-16 N-7 L-13 C-5	65
Failed to execute a step which we felt they meant to do but forgot	Forgot to set mos. Forgot duration Forgot print button Forgot to change one time field	K-3,44 L-82 C-7	4
Menu typo/select wrong command (Select wrong command or select same menu twice)		K-4, 23, 52 T-7 N-7,58 L-2,56,57,5960, 61,62,64,67,75	16
Meant to double click but moved mission/or meant to select mission and double clicked		K-15, 56 T-5,9,14 L-63,70,71,72	9
Time bar problems	It snapped back Wanted to stretch it and it slid.	T-11 L-68	2
Inadvertantly moved missions which involved repositioning executes		freq, tot # steps) T-3, (9) N-2, (10) L-5, (18) C-1, (9)	

Errors in Perception

Type	Instance	Subj/error #	Freq
Missions too close, overlapping	Many	T-4, N-24, 26, 27, 28 K-33	6

Dialog box already open		L-6	1
Doesn't notice changes in timebar	Time bar got bigger but it was not perceived.	L-50	1

Errors in Interpretation

Type	Instance	Subj/error #	Freq
Dates (schedule time period) not clear, difficult to interpret	Zulu time vs. EST	T-13	1
	What is last day to schedule	K-73	1
	Interprets schedule as being for 22 rather than 24 hours	L-69	1
Lack of feedback	Folders	T-56, K-74, L-29	3
	Find DB	T-57, 58	2
List select hard to distinguish between missions	Sorting Bravos	K-61	1
Poor field label - function unclear	Label in create form purpose - not realized changed name	T-49	1
Wrong input	Wrong date in find form (doesn't know it)	T-40	1
Memory	Forgot closed create mission screen	K-34	1
Screen format change unnoticed	Removing airspace layout and thought they were creating a folder	N-42,45	2
Scroll not greyed out	Thought scroll bar was active	N-61	1
Misinterpreted mission icon color meaning	Conflict state,	K-14,	2
	Denied state	L-76	
Misinterpret time bar	Already at beginning of week	N-14	2
	End of week	K-22	

Errors in Evaluation

Type	Instance	Subj/error #	Freq
System delay	Scheduling	T-38, N-23, K-17,19,20,21	6
Thinks all missions scheduled	Missions small	C-1	1
	Only monday scheduled	L-58	2
Scheduled mission out of time range		N-32	1
Wrong start time	Info. from previous mission remained in fields	L-34,40,41	3
Thinks there should be conflicts	No conflicts because did not approve any missions	L-52	
Aborts file menu for layout but wants folders		N-38	1
Lack of feedback	Changing airspace not making a folder	N-46	1
Misevaluate whether moving closer toward goal	Leaves date to set airspaces first	K-1	2
	Thinks items in a folder can not be found	K-69	
Memory	Aborts proper folder and edits another folder	K-29	1
Doesn't notice an execute error	Denied instead of approved, mission turned wrong color but didn't notice and thought was approved	L-66	1