

# INCREASED USER SATISFACTION THROUGH AN IMPROVED MESSAGE SYSTEM

Cassie L. Weissert  
Jet Propulsion Laboratory  
California Institute of Technology  
Pasadena, CA 91109  
(818) 354-4344  
Cassie.L.Weissert@jpl.nasa.gov

## ABSTRACT

With all of the enhancements in software methodology and testing, there is no guarantee that software can be delivered such that no user errors occur. How to handle these errors when they occur has become a major research topic within human-computer interaction (HCI). Users of the Multimission Spacecraft Analysis Subsystem (MSAS) at the Jet Propulsion Laboratory (JPL), a system of X and Motif graphical user interfaces for analyzing spacecraft data, complained about the lack of information about the error cause and have suggested that recovery actions be included in the system error messages. Existing research in the area of HCI design in relation to error messages was used to develop a standard for designing messages. This system extends the error messages to include more detailed and pertinent information, therefore enhancing user satisfaction. The system was evaluated through usability surveys and was shown to be successful.

## Keywords

Human-Computer Interaction, error handling, direct manipulation interface, usability evaluation

## INTRODUCTION

Why is it that in the age of 'user friendly', 'easy to use' computers that mistakes still occur [3]? As computers become a larger part of everyday life, the handling of errors also becomes more important. User errors may lead to a loss of productivity, and thus systems should be designed such that the user cannot make serious mistakes. However, designers should provide meaningful messages should they

occur. Systems therefore should contain simple, comprehensive mechanisms for handling these errors. Error checking and handling components may occupy the majority of a programming effort, but in most systems are left as a low priority. If error handling is designed properly, it can make the system appealing for the user. How to design error handling into a system has become a major research topic in the field of human-computer interaction design.

The Multimission Spacecraft Analysis Subsystem (MSAS) at the Jet Propulsion Laboratory is a set of software including procedures, models and utilities that provide a common spacecraft environment. MSAS development consists of three teams, the Engineering Analysis (or Downlink), the Uplink and the Database. The Engineering Analysis area is responsible for providing tools for analyzing spacecraft data sent down to earth and is the platform for this research. An Anomaly Report (AR) that exists on the system regarding error messages led to this research topic. The user wrote: "...require more descriptive error messages than MSAS provides, full explanation of the error cause, and suggested recovery actions be provided to workstation operators; these features are not adequately provided." This research examines a system for designing more detailed error messages into an interface without overloading the end user, hence leading to enhanced understanding, productivity and overall satisfaction. The types of messages will be described along with a methodology for building messages using this system. The results of the usability survey showing the systems success will also be addressed.

## MSAS DESIGN

The basic form of data within MSAS is called a State Table File. Each State Table File consists of multiple State Table Records. An individual State Table Record is a

representation of a single data value for a given channel id from the spacecraft at a specified instance of time. This data may be Predicted, Monitor (data from the ground antennae network), Quality Quantity and Continuity (quality and completeness of the data received), or Telemetry (downlink from the spacecraft). This value is referred to as the Record Type within a State Table Record. Two types of data values exist within a State Table Record, the Data Number (DN) and Engineering Units (EU). These data values may trigger three levels of alarms, red (critical), yellow (warning) or locally defined. The time related to this data value may be represented in up to four corresponding time formats, Earth Received Time (ERT), Spacecraft Event Time (SCET), Spacecraft Clock (SCIK) and Record Creation Time (RCT). These types are explained here due to their reference in many of the error messages.

Multiple tools exist within the Engineering Analysis subcomponent of MSAS for analyzing the downlink data and storing it into State Table files for subsequent processing. The State Table Editor, *steditor*, application plays a major role within MSAS as the main viewer and editor for these files after their creation. The *steditor* provides a tabular display of the State Table Records within the file. Each row in the table represents a State Table Record and the columns represent each of the data portions of the record. For viewing convenience the columns of the display can be locked into view, shifted, or hidden completely. The records may also be filtered by various data values in order to limit the number of displayed records to be viewed for analysis. The editor also allows for printing hardcopies of the display.

Record Type	Channel ID	Red Alarm State	Yellow Alarm State	Local Alarm State	Global Alarm State	EU Alarm State	DN Alarm State	EU Type
1	TELEMETRY	OK	OK	OK	OK	OK	OK	UNCORRECTED
2	TELEMETRY	OK	OK	OK	OK	OK	OK	UNCORRECTED
3	TELEMETRY	OK	OK	OK	OK	OK	OK	UNCORRECTED
4	TELEMETRY	OK	OK	OK	OK	OK	OK	UNCORRECTED
5	TELEMETRY	OK	OK	OK	OK	OK	OK	UNCORRECTED
6	TELEMETRY	OK	OK	OK	OK	OK	OK	UNCORRECTED
7	TELEMETRY	OK	OK	OK	OK	OK	OK	UNCORRECTED
8	TELEMETRY	OK	OK	OK	OK	OK	OK	UNCORRECTED
9	TELEMETRY	OK	OK	OK	OK	OK	OK	UNCORRECTED
10	TELEMETRY	OK	OK	OK	OK	OK	OK	UNCORRECTED
11	TELEMETRY	OK	OK	OK	OK	OK	OK	UNCORRECTED
12	TELEMETRY	OK	OK	OK	OK	OK	OK	UNCORRECTED

Figure 1: STEditor Screen Dump

The second application incorporated into the study is the State Table Compare, *stcompare*, tool. This tool allows for comparisons between two State Table files, often comparing the predicted to the actual values. The two files are displayed in tabular format in the main window using a specified time and data type. Options allow the user to customize the time and value tolerances to be used for each

channel that will be used in the comparison. After the comparison is executed, the discrepancies may be viewed in an ASCII formatted report. Statistics of the number of discrepancies found for each channel are also available. The *stcompare* application will be a very prominent analysis tool to determine if the spacecraft is performing as predicted.

Type	ID	SCET Time	DN Value
TELEMETRY	A-1010	1994-02-20 00:00:00	105296
TELEMETRY	E-0001	1994-02-20 00:00:00	31445
TELEMETRY	H-0004	1994-02-20 00:00:00	314159
TELEMETRY	E-0007	1994-02-20 00:00:00	345
TELEMETRY	A-1010	1994-02-20 00:00:00	105296
TELEMETRY	E-0001	1994-02-20 00:00:00	31445
TELEMETRY	H-0004	1994-02-20 00:00:00	314159

Figure 2: STCompare Screen Dump

MSAS uses Application Programmer Interfaces (API) for common library code. Two prominent APIs used throughout MSAS are the State Table and MSAS Time APIs. The State Table API provides the interface functions for creating and reading State Table files. The MSAS Time API defines the time routines used to validate and compare time strings as well as translate between the various time formats. Both the *steditor* and *stcompare* applications are based on the State Table file that includes all four time types and therefore both of these APIs figure prominently in their design. Due to encapsulation between the calling application and library routine, each API provides a routine for accessing the specific text for any error that occurs. For this reason, the proposed system will take into account the role of propagating these messages up to the calling applications.

## HUMAN-COMPUTER INTERACTION

The ACM Special Interest Group on Computer-Human Interaction Curriculum Development Group defines Human-Computer Interaction as

“the discipline concerned with the design, evaluation, and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them.”[3]

Human-Computer Interaction (HCI) is concerned more with the technical design of interacting with the machine itself rather than just the interface. Dix et al. define Human-Computer interaction as simply, "the study of people, computer technology and the ways these influence each other." [3] Ben Shneiderman states that HCI replaces arguments about "user friendly systems" with a more scientific approach. As a fairly new field, HCI combines the experimental methods and intellectual framework of cognitive psychology with the powerful tools from computer science [15]. As a true scientific field, human-computer interaction incorporates ergonomics, linguistics, and social science as well as computer science and psychology." The Handbook of Human-Computer Interaction states that HCI combines hardware oriented research in the areas of CRT screens and input devices and information presentation such as processing and cognition and user interface design [4].

Interface design improvements have led to reduced learning times, faster performance on tasks, lower rates of errors, and higher subjective user satisfaction according to Shneiderman [15]. Management is adopting user interfaces as a way to increase productivity, reduce fatigue and errors, and to enable users to be more creative in solving problems. Such a field is needed as the use of computers today is widespread and no longer limited to individuals with computer knowledge and experience. "User interfaces must be usable by the community at large and must serve the user's needs." [6]

## ERROR HANDLING

### Designing for Errors

Designing for maximum usability is the goal of interactive systems design. User-Centered Design is concerned with the interaction of the system as a cooperative endeavor: the task is not to find fault and to assess blame, but rather to get the task done [7]. Users perceive the computer as merely an aid in accomplishing their own job and resent messages that suggest the computer is in charge. Error messages result from the fact that the user has done something such that the system cannot respond. Both the computer and the user take the responsibility to repair the difficulty in misunderstanding each other as opposed to assigning blame. Lewis and Norman refer to this approach as "Let's Talk About It." They believe that much can be done to minimize the incidence and maximize the discovery of error while making it easier for the user to recover from the error [7]. The user-centered view emphasizes the user's intentions and actions, thus cognitive, behavioral, and social issues come into play.

A user-centered/user-support approach must consider the individuality and variability of users, growth and transition of user's skills, and the multitasking, improvisational, and error-prone nature of interaction [6]. Users are biased towards the path of "least cognitive resistance." This is the preference for mental data-processing strategies to read sequences of simple operations, fitting the task, and optimizing for transfer of previous results and minimizing the need for new information. Users make use of "established mental schema" formulated from previous encounters of a similar problem [6]. Designing for errors means providing user-recovery options to help the user deal with the effects of errors and feedback so that the user can detect errors immediately and clearly and understand the error made.

Human error is an essential topic for cognitive engineering if it is to impact performance. Error is the result of limited rationality -- people doing reasonable things given their knowledge. Error analysis consists of tracing the problem solving process to identify the point where rationality breaks down [4]. The term "human error" implies responsibility and blame focusing changes on local, incident-specific responses. In user-centered design the focus is on factors that produce the behaviors underlying disaster, emphasizing demand/resource mismatches. Human participation can prevent error prone points due to flexibility, adaptability and "intelligence" of the human [4]. Automation makes significant improvements, but post-conditions must be satisfied for the potential to be achieved and undesirable consequences to be avoided or mitigated.

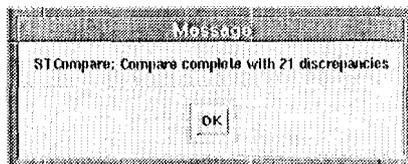
Runtime support is necessary as it is unrealistic to presume that all potential problems are identifiable before the interface is completed. The system must provide facilities that enable the user to deal with and manage errors as they arise [6]. Systems should be designed so that users cannot make serious errors. If an error occurs, the system should detect it and offer simple, comprehensible mechanisms for handling it. The advantage of increased productivity in direct manipulation interfaces may be lost due to user errors.

The general model of error-handling includes three steps -- detection, diagnosis, and correction. A user must first detect that a mistake has been made before they can correct it. Often a user knows that a mistake has been made immediately after or during the action itself, but in some situations external triggering such as on-screen messages may be required for detection. Diagnostic steps may also be used for stopping, possible mistakes before they occur through warnings such as "Name already exists. Replace?" when saving a file or "File has not been saved. Do you really wish to exit?". Diagnosis includes getting to know the nature of the mistake and the action(s) that led to it. Knowing the nature of the mistake can be vitally important

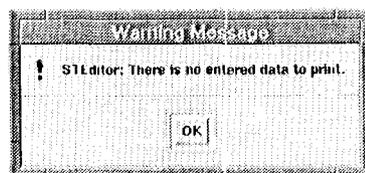
for chances of success in correction. The causes of the error are absent in many program messages. The third step, correction, includes goal setting, selection of a correction method, planning the execution method, and the physical execution of the corrective actions [5]. The presentation of error messages can assist the user in each of these areas.

### Classifications of Errors

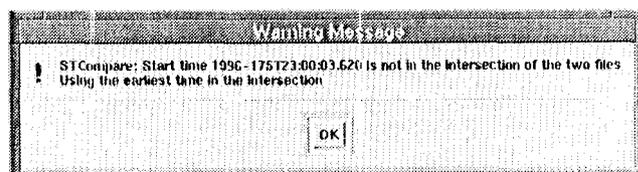
Feedback may be classified into three types: informative messages, warnings and errors. In MSAS, each is implemented using the standard Motif message dialog windows. Informative messages are used to present general information to the user. They may be used to display the current status of the running process or to state that a process is complete. In the State Table Compare application of MSAS, informative messages are used to convey the outcome of a comparison such as the following:



Warnings are used to tell the user when something inappropriate has or is about to take place. Warnings in the system protect the user from inadvertent destructive operations, yet allow the user to remain in control of the application. General warnings are often used to inform users that selected actions cannot be performed such as when the user selects the "Print" action when no data has been loaded.

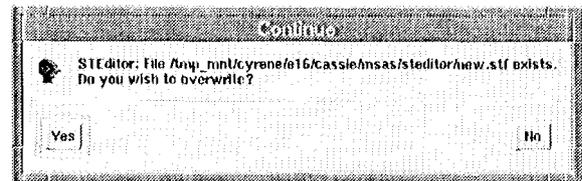


Warnings are also used to alert the users that default settings will be used, thus providing them with the chance to modify their selection before continuing.

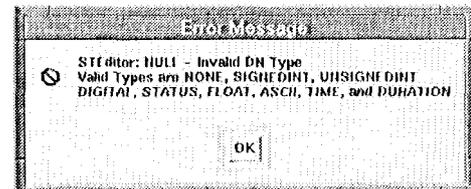


Warnings are often used to prevent a user from performing an action that may cause serious damage or unrecoverable loss of data. Warnings allow the user to choose between ignoring the warning and continuing or heeding the warning

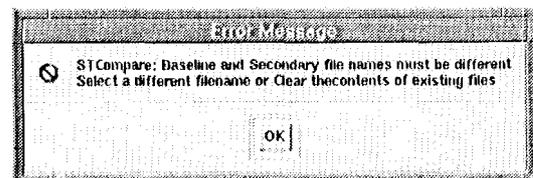
and canceling the operation. This type of warning is used to prevent corruption of previously saved data when a filename that already exists is selected for a Save operation.



The last class of feedback handles actual errors. Error messages provide feedback to the user when an invalid action has taken place or when illegal data values are entered. Error messages are accompanied by a single beep as is standard with the Motif error dialog. These messages are designed to help the user understand what is intended by the system in order to fix the problem so that the application may continue. Error messages should support recovery attempts and provide the information regarding the proper corrective action that is needed. To avoid excess errors, the controls may be temporarily disabled to allow the user to acknowledge and respond to the error. When illegal data values are entered by the user, a message containing the possible values that are accepted by the system can greatly increase the user's ability to understand and remedy the error.



A description of the recovery steps that can be taken to remedy an error situation are also extremely important, especially to new users,



User errors lead to loss of productivity and thus messages should tell the user specifically what the error is and what might fix it, using clear action verbs that emphasize the user's control of the program [1].

### Presentation of Error Messages

The presentation of the message can be as important to its content. The format and tone of messages are often offensive, especially for beginners, leading them to believe

that they personally have performed some serious offense [Inc. to their own] incompetence [7].

A system was designed to address the concerns of the users for more descriptive messages that includes an explanation of the error cause as well as suggested recovery actions. The format of messages adopted for use within this system is:

Program Window Name: Error Message  
Reason for Error or How to Recover

There are three components to this system for developing useful messages: traceability, tone, and specificity. Each component and its importance is described in the remainder of this section.

Information provided to users should be sufficient to allow them to find the problem, however most often this is not the case. This often occurs when multiple tasks are running at once. Errors often do not even say which program encountered the error so the user does not have any traceability to the routine in which the problem occurred, how to recover, or how to avoid the difficulty in the future. For this reason, the format studied by this system always states the name of the program or calling library routine in which the error occurred. This is especially important within MSAS where multiple applications may be running simultaneously, however it may be unnecessary in other systems where only one application is involved.

Messages should offer constructive guidance in a positive tone. Constructive messages are often difficult to apply since it is difficult to determine the user's intentions, but the user may be informed of the alternatives so that he can decide. Unfriendly messages include imperative commands, hostile messages, obscure messages and commands, as well as messages that are "too general" [9]. Imperative messages such as "Enter data Now" or "Wait, System busy" frustrate and confuse the user. A "Ready" type of message is preferable to one that commands the user. The intent of a user-centered system is for the user to feel that they are in control, not the computer. Hostile messages include those with words such as ILLEGAL, ERROR, INVALID or BAD. The use of the term "error" itself assigns blame to the user. Using "computerese" such as "JOB ill line. 14701" and "Improperly nested loops" heighten the fears of the user [9]. Messages that are designed to inform the user should inform them. Messages such as "wrong data" and "111<RROR47" are too general [9]. Many internet errors fall into this category. Errors such as "404 not found" and "403 Forbidden/access denied" are too general and uninformative. The message "Connection refused by host" is a better, less rude version of the 403

Forbidden error [2]. The user must consult the user manual to understand these general CI/IS.

Some examples of unfriendly messages and how they were improved helps to make these concepts clear. Changing the message 'ERROR 453 - NUMBERS ARE ILLEGAL' to 'MONTHS ARE ENTERED BY N/AM' provides a less threatening and more understandable message. This new message puts the user in charge while explaining what the problem is and how to make it right. Authoritarian messages like 'Enter next command' can be changed to 'Ready for next command' to make it more friendly and less offensive. A user-centered design would use messages like 'What do you need' in place of 'How can I help you' which emphasizes the computer's control. The message 'Syntax error' can be replaced with more specific messages such as 'unmatched right parenthesis'. Constructive messages and positive reinforcement produce faster learning and increase user performance [8].

Constructive messages provide the user with notification that detection of an error has occurred in addition to the steps to be taken to remedy the situation. Specific information relating to the proper format of requested data and where the problem arose should be included in the messages. Specificity of the provided messages is achieved by improving the clarity and consistency of the phrasing used. Variable names and program constructs that are unknown to the user should be avoided. Detailed messages containing information about the specific error and where it occurred provide the most specificity. These messages should also include suggestions as to how to remedy the situation.

Improvements have been made to 156 messages within the *steditor* and *stcompare* applications and library routines using the above system and have been subjected to user evaluation. The following examples should provide some insight into the understanding of the system.

The record type of the data is a required field. In previous versions of *steditor*, the message

*Invalid Record Type in row 5*

was given if it was missing. The new error message is

*STEditor: You need to specify a Record Type for row 5.*

This message provides more specific information about the need for the data, while also specifying which row was in error. A similar change occurred with the message

*Invalid DN Type in row 5.*

This error message was changed to

*STEditor: You must specify the DN Type for row 5.  
Use NONE for specifying 110 DN value.*

The new message adds the alternative solution to the user describing how to specify that no DN value be used.

The task of improving error messages was compounded by the use of Application Programmer Interface library functions. These errors are provided from the library and must be incorporated into the calling application. The previous version of steditor propagated the message

*Invalid MsasTime string*

to the application. The improved error message from the MSAS Time API is

*STEditor: Invalid MsasTime string format.  
Correct format is YYYY-DOYThh:mm:ss.mmm.*

The calling application is now referenced along with more information about the correct format that is expected. This system answers the complaints of the users by providing more information about the cause of the error in addition to information for fixing the problem. As a generic format, this system can be easily adapted to other

projects. The content of the messages in any system can be improved through the addition of a line containing more detailed reason and recovering information.

## EVALUATION

Two forms of evaluation were used to determine the extent of improvement made on the messages developed using this system within MSAS, analysis of error logs and subjective usability surveys.

### Analysis of Error Logs

Monitoring patterns of errors committed by users may help in further system design. Frequent errors may indicate changes needed to the system or may suggest modifications of the error messages themselves, changes to the command language, or the need for improved training procedures [8]. The evaluation of this system included the incorporation of an error log. Every time one of the two applications being improved was executed, an error log showing the user's name and each error message with the time at which it occurred was delivered through email. A total of 51 error logs containing a total of 21 messages from the system were analyzed from 10 different users. Table I shows these messages and the number of occurrences of each

Message Text	Number of Occurrences	Percentage
STCompare: Compare complete with <n> discrepancies	16	19.75
STEditor: Invalid record type	7	8.64
STEditor: <n> record(s) were not loaded due to errors "..."	7	8.64
Runstvaldate on the file to create an ASCII file for fixing the errors		
STEditor: Unable to read state table file: Invalid version of state table file	5	6.17
STEditor: You must select a row to delete	5	6.17
STCompare: There are no records containing you selected time/data	4	4.94
Choose different types from the option menu under the Session Settings panel		
STCompare: There are no records containing your selected data/time	4	4.94
Choose different types from the option menu under the Session Settings panel		
STEditor: Channel is Not Defined in Dictionary	4	4.94
STEditor: You must select a column to hide	4	4.94
STEditor: You must specify the DN Value for row <n> or use DN Type NONE	3	3.70

STEditor: Invalid Second Value in MsasTime String Valid values are between 00 and 59	2	2.47
STEditor: invalid MsasTime string format Correct format is YYYY-DOYThh:mm:ss.mmm	2	2.47
STEditor: You must select a row above which a new row will be added	2	2.47
STEditor: Invalid column to shift right	2	2.47
STEditor: invalid Alarm Value	2	2.47
STEditor: ASCII DN Values are maximum 12 characters long	2	2.47
STEditor: You must select n column to shift	1	1.23
STEditor: You need to select a column before locking	1	1.23
STEditor: There is no entered data to print	1	1.23
STEditor: You need to specify a Record Type for row <n>	1	1.23
STEditor: <n> records were not saved due to errors	1	1.23
STEditor: Unable to load file with Zero records	1	1.23
STEditor: invalid DN Type	1	1.23
STEditor: No records exist for filtering	1	1.23
STCompare Channel Settings Row <n> Invalid low tolerance value Low value must be a negative floating point	1	1.23
STEditor: Unable to convert binary to integer	1	1.23
<b>TOTAL</b>	81	

Table I: Error Occurrences in User Error Logs

The error logs indicated that two errors were often repeated during a single run of the application. The first of these is:

*STEditor: You must select a row to delete.*

The duplication of this message may indicate that the semantics for selecting a row are unclear to the user and that the message may need to be further improved to more explicitly describe this method. The second message was:

*STEditor: Invalid record type*

The values entered by the users included in these messages show that variations of **possible values were** attempted to see what was accepted by the application. The system has been further improved to allow the users to type only a limited number of characters for the record type and have the system fill in the remaining letters. For instance, the user may type only the character "T" and the system will fill in the remaining letters for the "Telemetry" record type.

Based on comments from the users, this will make the system easier to use and fewer typing errors will be encountered. The error logs used during testing, also provided a means of verifying the correct formatting of all messages.

#### Analysis of User Surveys

Subjective evaluation is an important component in the evaluation of interface usability. Evaluations can be used to measure the effect of the interface on the user and identify any specific problems. Ten subjects from the actual user population were used in this research. The subjects consisted of the three end users for these tools, three end users of uplink MSAS applications, the two testers for the project, one additional developer and the interface designer. These users have an average of 12 years of experience on a computer and 9 have at least college coursework in programming. The end users of these applications had over 100 hours of experience with MSAS while the users in the uplink areas of MSAS had only 20-25 hours using various

MSAS applications, and training experience only on the enhanced applications.

The subjects each completed the survey in my office. The survey began by running the two applications through a scenario that exposed them to eight different messages (5 error messages, 1 informational message and 2 warning messages) representing the enhancements to the system. A complete list of the enhanced messages was also available for their reference. Each of the subjects completed a post-session questionnaire that measured their attributions toward the system. The survey had multiple questions for comments and additional judgment ranks (on a scale of 1-6) for each of the areas of tone, tractability and specificity. The data from the questionnaires was analyzed and demonstrated that the users were very satisfied with the message system.

The traceability of the messages achieved an average score of 5.0 along with the appropriateness of the terminology used. The tone of the messages scored an average of 5.4 and the overall content scored a 5.8. '1111' users gave an average score of 5.83 for the helpfulness of the messages. Overall the messages were given an average score of 5.33. The users with the least experience on MSAS and no experience with the *steditor* and *stcompare* applications were hesitant to give higher than average scores after running only a small scenario, but all users had positive comments. The formal project User Acceptance Test report included the following comment regarding *stcompare*:

"The error and status messages are among the best of the MSAS V2 apps, especially for suggesting how to get out of the error state."

Further improvements may include a dual level message system. This system would provide general messages with the option for providing more details. This will address the concerns of one user surveyed that the messages are adequate for the experienced user, but the novice user may need more information. The operational workarounds for each of the messages may also be included in the help and program documentation with further details. These improvements will extend the system into a complete message system.

## CONCLUSION

Since the motivation of the research is to improve the user satisfaction of error messages with the *steditor* and *stcompare* applications of MSAS, development of improved error messages has been completed. The analysis of the

error logs helped to identify two messages that were unclear to the users and required further improvement. Individual user satisfaction was demonstrated through the results of the user surveys. Due to the success of the initial implementation, this system is now being implemented as a standard throughout MSAS and is being evaluated by other development groups at JPL.

## REFERENCES

1. Brown, J.R. and Cunningham, S. 1989. *Programming the User Interface Principles and Examples*. John Wiley & Sons, New York. 272 - 311.
2. Crowe, B. 1996. Anatomy of an Internet Error Message. *Computer Currents Interactive Magazine*. Computer Currents Publishing Corporation.
3. Dix, A., Finlay, J., Abowd, G., and Beale, R. 1993. *Human-Computer Interaction*. Prentiss Hall, New York.
4. Helander, M., Editor. 1988. *Handbook of Human-Computer Interaction*. Elsevier Science Publishing Company, Inc., New York.
5. Lazonder, A. W., van der Meu, H. 1995. Error-information in tutorial documentation: Supporting users' errors to facilitate initial skill learning. *International Journal of Human-Computer Studies*, 42, 185 - 206.
6. Lee, A. 1992. User Support: Considerations, Features, and Issues In *Advances in Human-Computer Interaction*. Edited by Hartson, H. R. and Hix, D. Ablex Publishing Corporation, Norwood, New Jersey. 184 - 221.
7. Lewis, C. and Norman, D. A. 1987. Designing for Error. In *Readings in Human-Computer Interaction: A Multidisciplinary Approach*, Edited by Baecker, Ronald M. and Buxton, William A. S. Morgan Kaufmann Publishers, Inc., Los Altos, California, 627 - 637.
8. Shneiderman, B. 1987. *Designing the User Interface Strategies for Effective Human-Computer Interaction*. Addison-Wesley Publishing Company, Reading, Massachusetts.
9. Smith, B. R. 1984. *Soft Words for a Hard Technology*. Prentiss-Hall, Inc., Englewood Cliffs, New Jersey. 28 - 32.