Error Messages: The Importance of Good Design

[redacted] STATUTORILY EXEMPT

Today's office environment is composed of distributed computer systems in which users accomplish tasks through interactive software packages. This has caused noncomputer professionals to become technically oriented to perform their jobs. Many users are finding that errors produced from interactive software packages are vague and require technical knowledge to understand and fix the problem. This paper discusses the problems caused by poor error message design and provides a set of guidelines and recommendations to help computer professionals design error messages for an interactive program.

INTRODUCTION

Anyone who has ever worked on a computer, whether a novice or expert user, will agree that error messages are a frustrating part of interacting with a computer. "Memory Fault: core dump" and "comsend dead write" are both examples of messages that confuse and frustrate many computer users. As computers become more dominant in the work environment, many noncomputer professionals are forced to become technically oriented to perform their jobs.

In the past, computers were centrally located and run by trained personnel. The users worked from listings provided by the computer support departments. Today, the office environment is composed of distributed systems where tasks are accomplished through interactive software packages. These interactive programs are fundamentally different from programs that produce a file or listing to be manipulated by the users at their convenience. In an interactive system, much of the work is done by the user, who is exchanging information with the computer.[1] Problems arise when the user has done something that the system cannot respond to, resulting in an error.[2] How the system tells the user that a problem has occurred is the key to program acceptance and user satisfaction.[3] If the error message is vague and confusing, the user will not have confidence in the program. This can result in low productivity, waste of man-hours and

1. James R. Williams, "Guidelines for Dialogue Design," *Designing and Using Human-Computer Interface and Knowledge Based Systems*, Proceedings of the Third International Conference on Human-Computer Interaction Boston, Massachusetts. Gavriel Salvendy and Michael Smith, gen. eds. Vol. II. (Amsterdam: Elsevier Publishing Company, September 1989), 589.

2. Clayton Lewis, and Donald Norman, "Designing For Error," *User Centered System Design: New Perspectives on Human-Computer Interaction*, eds. Donald A. Norman and S.W. Draper. (New Jersey: Erlbaum Associates, 1986), 411.

3. J.R. Brown, and S. Cunningham, *Programming the User Interface*, (New York: Wiley and Sons, 1989), 295.

resources, and low job satisfaction. Many problems caused by poor error messages could be avoided if system designers and programmers put more consideration into designing these messages for noncomputer-oriented people.

A27 is a good example of an office where nontechnical people must operate computers daily to accomplish their jobs. In the past three years, A27 has placed a computer on every analyst's desk in the hopes of increasing work productivity. Before this management approach was implemented, analysts worked from listings provided by a centralized computer department. Providing each analyst with a computer increased the amount and quality of work as long as the computer performed as the analyst expected. A major stumbling block that has confronted all A27 analysts is confusion over what the system is trying to "tell them" when a problem occurs. Even though more emphasis has been given to providing "user friendly" systems, analysts spend a great amount of time deciphering errors and fixing problems. The result is an increase in wasted man-hours and resources and a decrease in the amount and quality of work being produced. The A27 computer team spends most of its time helping analysts with computer problems. Many of these problems are caused by bad error message design.

WHY ARE ERROR MESSAGES DESIGNED SO POORLY?

The answer to this question lies in how a system designer and programmer defines and perceives an error. An error is defined simply as an action being performed by a user who does not enter the request in a form the computer can understand. Lewis and Norman state "...what is really meant is the system can't interpret the information given to it. By convention, this is called an error, a fault generated by the user."[4] User error is not the only kind of error that can occur during execution of an interactive program. Errors are divided into four classes: user input errors, computational errors, file errors, and device errors. User input error is the most common and hardest to control. These errors are caused by faulty control input such as wrong choices, mistaken commands, and invalid data.[5] User input errors are divided into slips and mistakes. "A slip is defined to be an error where the action that is done is not what the user intended."[6] For example, when deleting a file, the wrong name is specified. "A mistake is defined to be an error where it is the intention that is wrong."[7] For example, a user deletes a file because he thought it would no longer be needed. Mistakes result from not diagnosing the situation correctly. For the user, these are harder to discover than a slip. When a slip occurs, discovery is quicker because the action that was performed is different from the action that was intended.[8]

4. Lewis and Norman, "Designing for Error," 412.

5. Brown and Cunningham, *Programming the User Interface*, 295.

6. National Cryptologic School Satellite TV, "User Centered System Design," *User Interface Strategies '88*, material prepared for the University of Maryland Instructional TV, 28 July 1988, Donald Norman, 7.

7. Ibid.

8. Lewis and Norman, "Designing for Error," 419.

The other three classes of errors normally occur from within a program and are easier to control. Computational errors are attempts to do impossible arithmetic such as dividing by zero.[9] Improper uses of computer devices are classified as device errors, while file errors are illegal operations on a file.[10]

Even though each type of error is generated from different sources, all errors normally produce a message and require some response from the user. Many designers and programmers do not take into account that these messages are being designed for a person and that human beings are not perfect. A system that requires a person to perform correctly all the time will be frustrating to the user. "The system is just as much to blame for its failure to understand as it is for the user to be perfectly, unambiguously precise."[11] Attempting to create a system that understands all the user's intentions is also unrealistic. However, a system can be designed to handle many errors gracefully. How a program handles errors and tells the user what is wrong can be the key to user acceptance of the program.[12] The more understandable a program is, the more use the program will get. Errors should not be perceived as the fault of the user or something to avoid. Think rather that errors are the natural result of a user accomplishing a task and that the system should do its best to inform the user where the problem is and how to solve it.[13]

HOW CAN ERROR MESSAGES BE DESIGNED TO INCREASE PRODUCTIVITY AND USER SATISFACTION?

The following eight guidelines can help designers and programmers write effective error messages:

Design error messages for the intended users.

Write error messages that are specific and constructive.

Avoid anthropomorphism.

Write error messages using a positive tone.

Put the main idea first.

Make error messages timely.

Be consistent in the grammatical form, terminology, abbreviation, and visual format and placement.

Provide multiple levels of error messages.[14]

9. Brown and Cunningham, *Programming the User Interface*, 295.

10. Ibid.

11. Lewis and Norman, "Designing for Error," 414.

12. Brown and Cunningham, *Programming the User Interface*, 273.

13. Lewis and Norman, 431.

14. National Cryptologic School, "Lesson Four," *MP-328: User Interface Design*, (Maryland, National Cryptologic School, 1990).

Designers and programmers need to take into account that the user is not just an input device but is an integral part of the interactive system who is apt to make errors along the way. Taking into consideration that error messages are intended for people is the first guideline in designing more productive error messages. As in writing a paper, the author should define his audience and the level of knowledge of that audience. The error message should also be centered on the task the user is trying to accomplish. In most cases, a user of an interactive system will be a nontechnical person trying to accomplish some task. Therefore, the designer or programmer should put himself in the user's shoes, designing error messages addressing both the intended audience and the task at hand. Grudin maintains that "users and their tasks should be the overriding consideration in user interface design."[15]

The two error messages in the introduction of this paper, "Memory Fault: core dump" and "comsend dead write," demonstrate a significant problem with error message design. That is, they are too vague and confusing to users, including computer-literate users. Error messages should be designed to be as specific and constructive as possible. When insufficient information is provided, the user wastes time understanding the problem and developing a solution. This can lead to user frustration and dissatisfaction.[16] A good error message should tell the user precisely where the problem is and what might be done to correct it.[17] The following example demonstrates this guideline:

POOR DESIGN:	cannot open input file
BETTER DESIGN:	/home/termfile is not found: check spelling.
BETTER DESIGN:	/home/termfile is not found. Maybe typing error. Enter filename or return to exit:
BETTER DESIGN:	/home/termfile is not found. Did you mean /home/termsfile? Enter a filename, y to use /home/termsfile, or return to exit:

The error message "cannot open input file" violates the second guideline to writing effective error messages because it does not tell the user exactly what the problem is or how to fix it. It does not even give the name of the file in question. The user may not be able to open the file because the file does not exist or the user does not have open privilege. A more productive interface would determine why the file could not be opened and state this in the error message. An even better interface would not only tell the user the problem but help correct it.

The third guideline in writing effective error messages is to avoid anthropomorphism, which is assigning human characteristics to inanimate objects. Messages that sound as if

15. J. Gordin, "The Case Against User Interface Consistency," *Communication of the ACM*, Vol. 32, No. 10 (October 1989), 117.

16. Ben Shneiderman, [Closing Plenary Address] *Seven Plus or Minus Two Central Issues in Human-Computer Interactions*, (Human-Computer Interactive Laboratory: University of Maryland, April 1986), 106-7.

17. Brown and Cunningham, 309.

the computer is an independent being normally contain words that will distract the user from the meaning and intent of the message. "Dialogues should not contain information that is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility."[18]

Writing the error message in a positive tone is just as important as the message content. "The way you tell a user something is going wrong has a large effect on his perception of the program."[19] A harsh error message only frustrates and confuses the user. This leads to reduced job performance and productivity. For an interactive program to be effective and useful, the user needs a well-written error message that does not make him feel incompetent and unskilled.

In any form of communication, it is important that the main idea be first. A person should not have to waste time trying to understand the idea being communicated. As with interactive programs, the user should recognize from the beginning that the message is communicating a problem. The "file not found" examples demonstrate this fifth guideline by informing the user that the file entered cannot be found. The last two design examples also inform the user that the program will exit if a correct filename is not provided.

A designer or programmer should always make error messages timely. "Detecting error is the first step to recovery. Early detection of error is extremely important. Delayed detection obviously wastes time and effort as the user works through steps that will not be effective because of an earlier error."[20] Informing the user that an error has occurred when it is encountered reduces frustration and fatigue. It keeps the user from having to back up to that part of the task where the error occurred. By making the error message timely, the user is still focused on the part of the task that is having problems. UNIX is a good example of an interface that waits until after the user has entered all parameters to a command before it tells the user that a parameter is invalid. The user wastes time determining which parameter is wrong and reentering the command. If the user makes another mistake when reentering the command, the cycle is repeated. In an attempt to reduce typing errors, some UNIX systems provide users with a history log that contains a list of commands executed during a session. Commands in this log may be edited and reexecuted. In many cases, the user must still determine why the command failed.

Error messages should be consistent in grammatical form, terminology, abbreviation, and visual format and placement. This seventh guideline is centered on how people learn and process information. The human mind is composed of long- and short-term memories. The long-term memory contains information learned throughout life while the short-term memory contains small amounts of information needed for only a limited amount of time.[21] People function more efficiently when the task conforms to knowledge already stored in

18. Jakob Nielsen, "Traditional Dialogue Design Applied to Modern User Interfaces," *Communication of the ACM*, Vol. 32, No. 10 (October 1989), 111.

19. Brown and Cunningham, 306.

20. Lewis and Norman, 419.

21. Donald Norman, "The Trouble with Unix," *Datamation*, Vol. 34, No. 12 (November 1981), 150.

memory. A user has an easier time solving problems when recognizing something previously learned than having to retrieve the information without any prompts or aids.[22] By making error messages consistent with the user language, the amount of time required to understand and solve a problem is greatly reduced. Therefore, error messages should be expressed in words, phrases, and concepts familiar to the user. Do not use system-oriented terms if the users are nontechnical people. "Users should not have to wonder whether different words, situations, or actions mean the same thing."[23] By following a consistent format and placement, error messages are more readily visible to the user. Consistency is the most important guideline to follow when designing error messages because it allows the user to draw from past experiences to solve problems. This reduces the amount of time in problem solving and increases both work performance and job satisfaction.

When designing error messages in an interactive program, the designer and programmer must consider the different levels of knowledge and experience of the users.[24] Novice users are new to the computer environment or a particular software application. They need help frequently. Casual users have experience but do not use the computer or software application regularly. These users may need help occasionally. The frequent user is very skilled with the computer environment or the application programs. These users can handle many problems without much help or guidance. Error messages need to be designed to meet the needs of each type of user. Error messages that do not provide enough information to help novice and casual users will be a source of frustration. The frequent user may become annoyed when too much information is provided. Therefore, an interactive program should provide multiple levels of error messages. Many applications repeat the same error message until the user enters a correct response. After the second or third incorrect response, the user apparently does not understand the problem or how to solve it. Instead of keeping the user in a cycle that is not solving the problem, the designer or programmer should display more detailed error messages for each incorrect response. By providing multiple levels of error messages, novice and casual users can be helped without hindering the frequent users. The following example demonstrates this last guideline:

```
PROMPT:    Enter the file to be searched: term <return>

LEVEL 1:   /home/term is not found. Did you mean /home/terms?
           Enter a filename, y to use /home/terms, or return to exit: term1 <return>

LEVEL 2:   /home/term1 is not found.

           List of files in /home directory:
           1: db1file   3: terms    5: terms2   7: termsfile
           2: db2file   4: terms1   6: terms3
```

22. J. Karat, "The Relation of Psychological Theory to Human-Computer Interaction Standards," *Designing and Using Human-Computer Interface and Knowledge Based Systems*, Proceedings of the Third International Conference on Human-Computer Interaction, Boston, Massachusetts. Gavriel Salvendy and Michael Smith, gen. eds. Vol. II. (Amsterdam: Elsevier Publishing Company, September 1989), 584.

23. Nielsen, 111.

24. Brown and Cunningham, 273.

To choose a file, enter the number or filename

To list another directory, enter directory name

To exit, enter return: 4 <return>

Each incorrect response provides more detailed information until the problem is solved or the user exits the process. The frequent users are not hindered with detailed and lengthy messages that, nevertheless, are beneficial to the novice or casual users.

WHAT CAN COMPUTER PROFESSIONALS DO TO ENSURE BETTER DESIGN OF ERROR MESSAGES IN INTERACTIVE SOFTWARE PACKAGES?

The following recommendations can help software developers improve the quality of error message design in projects:

Increase attention to error message design.

Develop error message guidelines for projects.

Design, review, test, and revise error messages.

Conduct user acceptance testing.[25]

To increase the quality of error message design in software products, designers and programmers must be made aware of the importance of error messages in interactive programs. Poorly designed error messages cause users to waste time understanding and fixing problems. The results are low productivity, waste of man-hours and resources, and low job satisfaction. Emphasis must be placed on the role of error messages in accomplishing a task. Errors should not be considered an exception but part of the normal operation of a program.[26] "Think of each action by the user as a step in the right direction: an erroneous action is simply one that is incompletely or improperly specified. Think of the action as part of a natural, constructive dialog between the user and the system."[27] By increasing attention to error message design, attitudes about the role of error messages in interactive programs can be changed to improve the quality and increase the helpfulness of error messages for the user.

Designing software is no longer done when the programmer begins to write the code. Instead, many organizations have a software development plan or system life cycle. This cycle breaks the analysis, design, implementation, evaluation, and maintenance into separate steps to increase quality and productivity of software. Software products developed under this methodology contain a set of guidelines documents that record the purpose and design rules. Providing a set of written guidelines for error message design is the second approach to increasing the quality of error messages in software products. The

25. National Cryptologic School, "Lesson Four," *MP-328: User Interface Design*, (Maryland, National Cryptologic School, 1990).

26. Lewis and Norman, 431.

27. Norman, 15.

guidelines document not only records the rules for designing error messages but also stresses the importance of error message design. For the guidelines document to be successful, it should allow exceptions when a good justification is given. "Successful guidelines documents have multiple levels of strictness: rigid standards that cannot be violated, strict guidelines that require management approval for an exemption, or softer practices that can be violated if the designer finds good justification."[28]

Designing error messages should also follow the system life cycle approach. Review, test, and, when appropriate, revise the error message after the initial design. Authors do not publish their first draft; therefore, programmers should not accept the first design as the final product. Placing error messages in a separate file increases their availability for reviewing and revising.[29] This not only makes it easier to check grammar, terminology, spelling, and consistency but allows other programs to reuse the error messages, reducing development time and cost.

"Testing is no longer seen as the last minute validation of a design but has become the normal process for refining ideas from the beginning."[30] By testing the software throughout the design process, computer professionals can find mistakes and correct them as early as possible. It is more cost effective to find and correct problems at the beginning of a design than after the final version has been released. For testing to be efficient, it must be done by the people who are to use the final version of the product.[31] The last recommendation is to set up user acceptance testing for error messages. Allowing users to test error messages during the design process not only will increase the quality but will help designers and programmers keep their focus on the intended audience.

CONCLUSION

Error message design is just as important as the design of the overall interface. However, very little attention is given to it. This is evident in many interactive programs. These programs often display error messages that are vague and confusing to the people who operate them. Designers and programmers do not take into account that error messages are intended for users who have different levels of knowledge and experience with computers. Error messages should be viewed as a form of help for the user. They should state in terms understandable to the intended audience what the problem is and how to solve it.

The two error messages in the introduction of this paper demonstrate how error messages should not be designed. "Memory Fault: core dump" is produced from a "C"

28. Ben Shneiderman, "We Can Design Better User Interfaces: A Review of Human-Computer Interaction Styles," *ERGONOMICS*, (Massachusetts: Addison-Wesley, 1988), 61.

29. Shneiderman, [Closing Plenary Address] *Seven Plus or Minus Two Central Issues in Human-Computer Interactions*, 107.

30. Shneiderman, "We Can Design Better User Interfaces: A Review of Human-Computer Interaction Styles", 61.

31. Norman, 9.

program used in A2. This error is generated when the input file contains one line. Even though this line is valid, the program reads past the end of the file producing this message. By inserting a blank line, the program operates with no problem. The program was designed to operate on two or more lines. However, the programmer neglected to produce a correct response for the user. No reference in the documentation states how the file should be formatted.

"Comsend dead write" is another example of a badly designed error message. The message is produced from an interactive program that transfers files from one system to another. No one seems to know what this message means except that the transfer was unsuccessful. The user must retry the program until the transfer works. In both cases, the user and the computer support team waste time and resources solving these problems. The user wastes considerable time wondering what went wrong and trying to correct it. When the user cannot solve the problem, the user's computer support team is contacted. Again, time is wasted by the team trying to understand what the program does and why it failed. Resources are wasted from calling the office that created the software and retrying the program until a solution is found. The longer it takes to determine the cause of the problem, the more the user becomes frustrated with the system. This leads to a decrease in productivity and job performance.

Computer professionals can improve the quality of error messages in interactive programs by increasing attention to error message design, developing error message guidelines, and working with users to review, test, and revise the design. As Shneiderman stated, "improvements in error messages are easy to make and provide measurable benefits to the users."[32]

32. Shneiderman, 107.

STATUTORILY EXEMPT

[redacted]

BIBLIOGRAPHY

Brown, J. R., and S. Cunningham. *Programming the User Interface*. New York, New York: Wiley and Sons, 1989.

Grodin, J. "The Case Against User Interface Consistency." *Communications of the ACM*. Vol. 32, No. 10 (October 1989), 1164–73.

Karat, J. "The Relation of Psychological Theory to Human-Computer Interaction Standards." *Designing and Using Human-Computer Interface and Knowledge Based Systems*, Proceedings of the Third International Conference on Human-Computer Interaction, Boston, Massachusetts. Gavriel Salvendy and Michael J. Smith, gen. eds. Vol. II. Amsterdam: Elsevier Publishing Company, September 1989.

Lewis, Clayton, and Donald A. Norman, "Designing for Error." *User Centered System Design: New Perspectives on Human-Computer Interaction*. Eds. Donald A. Norman and S. W. Draper. New Jersey: Erlbaum Associates, 1986.

National Cryptologic School. (Lesson Four) *MP-328: User Interface Design*. Ft. George G. Meade, Maryland: National Cryptologic School, 1990.

National Cryptologic School Satellite TV. "User Centered System Design." *User Interface Strategies '88*. Material prepared by Donald Norman for the University of Maryland Instructional TV. 28 July 1988.

Nielsen, Jakob. "Traditional Dialogue Design Applied to Modern User Interfaces." *Communications of the ACM*. Vol. 33, No. 10 (October 1990), 109–17.

Norman, Donald A. "The Trouble with Unix." *Datamation*. Vol. 34, No. 12 (November 1981), 139–50.

Shneiderman, Ben. [Closing Plenary Address] *Seven Plus or Minus Two Central Issues in Human-Computer Interactions*. Human-Computer Interaction Laboratory, University of Maryland. April 1986.

Shneiderman, Ben. *We Can Design Better User Interfaces: A Review of Human-Computer Interaction Styles*. Reading, Massachusetts: Addison-Wesley, 1988.

Williams, James R. "Guidelines for Dialogue Design." *Designing and Using Human-Computer Interface and Knowledge Based Systems*. Proceedings of the Third International Conference on Human-Computer Interaction, Boston, Massachusetts. Gavriel Salvendy and Michael J. Smith, gen. eds. Vol. II. Amsterdam: Elsevier Amsterdam: Elsevier Publishing Company, September 1989.