ARTICLE

Help texts vs. help mechanisms: A new mandate for documentation writers

NATHANIEL SOLOMON BORENSTEIN, Carnegie Mellon University, Pittsburgh, PA, United States

Published: 01 February 1986

SIGDOC85: Fourth International Conference on Systems Documentation
June 18 - 21, 1985
New York, Ithaca, USA

Conference Sponsors:
SIGDOC

SIGDOC '85: Proceedings of the 4th annual international conference on Systems documentation (February 1986)
https://doi.org/10.1145/10563.10576
ISBN: 0897911865

Help Texts vs. Help Mechanisms:

A New Mandate for Documentation Writers

NATHANIEL S. BORENSTEIN

Information Technology Center
Carnegie-Mellon University

Abstract

To compare different methods of accessing and presenting on-line help, controlled experiments were conducted. Several different help systems were compared, including a natural language help system and a human tutor. The results indicate that, while varying the help mechanism may have some effect on learning, its importance is greatly overshadowed by the simple quality of the help texts being presented. Even for on-line help, good writing seems to be the most important part of helping the user, far more important than elaborate or sophisticated mechanisms.

1. Introduction

Historically, little attention has been paid to on-line help mechanisms. They have been built, if at all, as afterthoughts, last-minute additions to complex software systems. In recent years, however, they have begun to be addressed more seriously. Several authors have made surveys of on-line help systems [1, 10, 17, 19] or have made genuine attempts to carefully design such systems [3, 5, 6, 7, 9, 12, 13, 14, 20]. Though these surveys and design projects have greatly helped to define the state of the art in on-line help, they have provided precious little hard data about the usefulness of on-line help itself, nor about the genuine advantages and disadvantages to the various mechanisms that have been used. Only Magers [11] actually conducted controlled experiments, which studied the effect of simple changes in a help system *and* the underlying application interface. Yet recent developments in evaluating user interfaces, such as the Roberts and Moran methodology for evaluating text editors [2, 15, 16] have made it clear that such evaluation is both possible and desirable.

The experiments discussed here were conducted as part of the author's doctoral dissertation on on-line help systems [1]. The goal of that research was to experimentally evaluate the state of the art in help systems, and to try to find out which qualities make a help system most useful.

The research consisted of three phases. In the first phase, a thorough survey of previous work on help systems was conducted, and extensive surveys and protocols were taken of actual users of existing help systems.

From these explorations, a prototype help system was designed in the second phase of the research. This system was designed to fairly approximate the state of the art in help systems; no existing help system was used because no single system was available which incorporated enough of the features which had been found in *some* help system. That is, no help system could be found that availed itself completely of even those techniques which had been documented in the published literature. The prototype system, known as ACRONYM, will be described briefly in Section 2.

In the third phase of the research, controlled experiments were conducted to compare the prototype help system with other help systems and learning conditions, with surprising results. The experimental methodology and the most relevant results of the experiments will be described in Sections 3 and 4.

2. The ACRONYM Help System

The prototype help system designed for these experiments was called ACRONYM. ACRONYM was a help system for UNIX, which knew about most of the UNIX[1] utility programs. ACRONYM's mechanisms are described extensively elsewhere [1], but a brief description will be provided here.

ACRONYM ran on a VAX 11/750, running UNIX and Emacs [8]. As a terminal, it used a Xerox Alto computer running a terminal emulator, which gave a 60 line screen and supported pointing with a mouse. ACRONYM ran with the user's screen divided into three windows, as pictured in Figure 1. The top window was the "Help texts" window, in which the system displayed the actual help texts. The second window was the "Help menu" window, in which the system provided a menu of topics for further help requests. The bottom window was a "Commands" (shell) window, in which the user actually typed commands and viewed their output. Because a 60-line terminal was used, each window was nearly as large as a standard video display terminal's screen.

ACRONYM provided help via four basic mechanisms. First of all, it provided context-dependent help without any initiative on the user's part. Each time the user pressed the SPACE key, to separate words of his command, the system automatically updated the two help windows to reflect the user's new context. Thus, if the user typed "ls", the UNIX command to list the files in a directory, ACRONYM would put an explanation of the function and options of the "ls" command in the top window, and a menu of related topics in the middle window.

[1] UNIX is a trademark of AT&T Bell Laboratories

Second, ACRONYM allowed users to invoke further context-dependent help by pressing a single key, "?". This would give help in a manner similar to the automatic help provided when SPACE was pressed, but often more specifically geared to the current context. (This help included, for example, file name completion, so that a user could type part of a file name, press "?", and see the possible completions of that name.)

Third, ACRONYM provided menu help using a mouse. Users simply pointed at any item in the menu window and pressed any key on the mouse, and the help and menu windows were updated accordingly.

Finally, ACRONYM provided key word help. Users could, at any time (even in the middle of a command line) type "help", followed by a key word, and ACRONYM would update its help window and menu window in accordance with the key word. If the key word was unambiguous, (e.g. "help ls"), the appropriate help would be displayed in the help text window and a menu of relevant items would appear in the menu window. If the help request was ambiguous, a message explaining this fact appeared in the top window, and a menu of possible interpretations would appear in the menu window. Thus, the intended topic could usually be selected with the mouse.

ACRONYM's database was designed so that most texts and menus would fit in the windows provided. However, when this was not the case (for example, "help file" provided a relatively long list of related topics in the menu window), scroll buttons were provided on the screen, so that each window could be scrolled independently backwards or forwards by pointing with the mouse.

Texts used in the ACRONYM database and in the hybrid system described below came from the book *A Practical Guide to the UNIX System* by Mark Sobell, and were reproduced with the consent of the author and publisher [18].

3. The Experimental Method

In each of two parallel experiments, a group of subjects with similar backgrounds and experience in using computers were given a set of tasks to perform on UNIX. In one experiment, none of the subjects had ever used UNIX before, but all had performed similar tasks on the TOPS-20 operating system. The second experiment studied UNIX experts; naturally they were given a different set of tasks.

The independent variable in each experiment was the method by which the necessary help information was obtained. Each subject used the standard CMU UNIX help system (see below) during half the experiment, and used one of the other help methods during the other half. (The use of the standard system as a *baseline* condition was designed to reduce the effects of subject variation.) These were balanced so that an equal number of people used each of the non-standard help systems for each half of the experiment.

The dependent variable measured was the time it took to successfully execute each task. The experiments were videotaped, and times were computed from the time stamp on the videotape.

In order to limit the time of the experiment and to insure that no subject got bogged down with a single task early in the experiment, a cap of ten minutes was placed on task execution time. The tasks were small enough that this was enough time for nearly all subjects on nearly all of the tasks. (The task selection and the ten minute cap were the result of a series of earlier pilot experiments designed to select tasks of appropriate size and difficulty.) When a subject failed to complete a task in ten minutes, the experimenter showed him the right solution (the right way to get the task done) and then allowed him to go on to the next task.

There were 22 tasks, divided evenly into two comparable sets. At the midpoint of the experiment, the subjects were shown a different way of getting help, and were required to use that second method during the second half of the experiment. The task order was fixed throughout the experiment; the nature of the tasks themselves imposed at least a partial ordering, making it difficult to vary the task order in any reasonable way. A summary of the tasks for the experiments is given in Table 1.

Five different help systems were studied. These are summarized in Table 2, and described in detail below.

3.1. The "Baseline" Help System: man and key

The "baseline" help system, H_0, which each subject used for either the first or second half of the experiment, is the standard help system used on the CMU UNIX systems. This system consists of two commands, ***man*** and ***key***. The man command is used to print the complete UNIX manual entry for a given command. The key command can be used to find out about unknown commands; users type "key file", and the system prints a single descriptive line for each manual entry that it finds for the key word "file". (On most UNIX systems, the "key" command is absent, but a similar "apropos" or "man -k" command often exists.)

This system has several problems. First, the texts are of extraordinarily poor quality, by almost any standard. Second, the key word lookup is done in a very stupid manner: a key word matches a manual entry only if the word is an exact substring of the first line of that manual entry. Third, the man command, for printing out manual entries, is very slow because it runs the entire manual entry through the nroff text processing utility before printing it out.

Subjects using the baseline system were supplied with a physical copy of the UNIX manual, so that they did not actually have to sit still and wait for the man command to perform. They were also supplied with a booklet titled "UNIX for Beginners", which is generally supplied as part of the standard UNIX documentation for new users.

3.2. The Hybrid System

The second help condition studied, H_1, was a hybrid system that consisted of the same mechanisms used in the standard system (H_0), but with better texts. (These texts were derived from the ACRONYM help system described in Section 2, and hence came in large part from Sobell's book [18].) The mechanisms were the same as the standard system at the user level, but performed better -- the man command was faster, and the key command, though somewhat slower, did a much more thorough search for key words. This hybrid system is thus best thought of as the standard system "done right." Users of the hybrid system received exactly the same instruction sheet and supplementary materials that were given with the baseline system. Of course, the paper copy of the manual which was given to these subjects contained the improved, non-standard texts.

3.3. The ACRONYM Help System

Help condition H_2 was the ACRONYM prototype help system, described in Section 2.

3.4. The Human Tutor

Help condition H_3 was a human tutor. Subjects with this help condition were allowed to ask any question of the tutor, but were not allowed to rely on the tutor's prior knowledge of what the problem was. Hence, all they had to do was to state the problem clearly and in their own words in order to have the solution explained to them. When in doubt, the tutor tried to consistently

err, if at all, on the side of being too helpful, so that this help condition may be regarded as an "idealized" human tutor.

3.5. Simulated Natural Language Help

The final help condition studied, H_4, was a simulated natural language help system. Subjects with this help condition were allowed to ask any question in natural language by typing it on their keyboard; the responses were determined by the experimenter in the next room, whose participation was not known to the subjects and came as a surprise to all of them when the deception was revealed after the experiment. Special support software allowed the experimenter to react quickly to each help request by sending the user a small portion of the ACRONYM database; thus the experimenter acted as an English-to-ACRONYM translator. As with the human tutor, the translator tried to err, if at all, in the direction of being too helpful.

4. Experimental Results

The experiments yielded a highly complex set of data, due to the fact that there were differences due not only to help system variation and individual subject variation, but also to the variation of task difficulty. (Some of the tasks took less than a minute to complete, on average, while others took several times as long.) In order to obtain statistically significant results, a regression analysis was used, the full details of which are reported in the author's dissertation [1]. Here, the results will be summarized but not detailed.

4.1. Novice Results

The novice experiment, as expected, showed the human tutor to be the best help system, and the standard (baseline) system to be the worst. The other three systems -- ACRONYM, the hybrid, and the simulated English help system -- all performed about equally well, about halfway between the other two systems. The difference between the standard system and these three, and the difference between these three and the human tutor, were reasonably significant ($p < .05$ for most of the differences, slightly higher for the difference between ACRONYM and the tutor).

Figure 2 shows the distribution of novice timings for each help system. While this graph makes it seem likely that finer distinctions can be made -- e.g. that the natural language system is better than ACRONYM, and that ACRONYM is slightly better than the hybrid -- the data does not support these conclusions with a high level of significance.

Although some second-order differences are apparent from a task-by-task analysis (not included here) -- for example, ACRONYM appears to perform best on the hardest task, and to actually perform extremely poorly on the simplest tasks -- one overall result is striking: the three systems that use the same good help texts perform nearly equally well, far surpassing the simple baseline system. Indeed, just comparing the baseline system to the hybrid shows that by improving the texts and indexing, nearly half the time difference between the standard system and an *idealized* human tutor was eliminated.

4.2. Expert Results

In the expert study, the natural language condition was not tested. Of the remaining four help systems, once again the human tutor performed very well and the baseline system relatively poorly, although the difference was not as great as with the novices. Like the novices, the experts responded very well to the improved texts in the hybrid system -- so well, in fact, that there was no significant difference between the human tutor and the hybrid help system!

The experts' response to ACRONYM was very different, however. Using ACRONYM, the experts performed quite poorly. In fact, expert performance with ACRONYM was not significantly better than their performance with the standard (baseline) help system.

This result, though at first surprising, seems to have a simple explanation. The UNIX experts studied were people who have been using man and key for years. These commands are second nature to them. (In fact, Draper [4] has suggested that familiarity with the help system is the only reasonable criterion for assessing expertise in a system like UNIX.) By improving the texts and indexing without changing the mechanism, the hybrid system allowed them to leverage their previous knowledge, and to perform spectacularly well. When new access mechanisms were substituted for the old, as in ACRONYM, the benefit of the new texts was offset by the hurdle of requiring the learning of a new help system.

The distribution of Expert times is shown in Figure 3.

5. Conclusions

The most important result of these experiments is support for the claim that quality of writing is the most important single aspect of on-line help systems. This is, of course, something which system documenters have believed for a long time, but it has been difficult to convince the technically-minded of this belief. The simple fact, however, is that simply improving the text and indexing of the standard UNIX help system yielded a tremendous improvement in the rate at which novices learned to perform a fixed set of tasks. (Very roughly, they averaged 1 minute per task with a tutor, 2 minutes per task with the improved texts, and 3 minutes per task with the standard system.)

These results should not be interpreted as saying that help mechanisms are not important. For one thing, the data suggested -- but could not prove -- that the more sophisticated mechanisms did provide small improvements in learning times, especially on the hardest tasks. Moreover, a subjective evaluation, not reported here, indicated that users generally preferred the help mechanisms in ACRONYM, even if those mechanisms didn't really improve performance. Since it is such subjective impressions that sell software, this is of no small importance. Nonetheless, these experiments do indicate that anyone who wants to build a really good help system can not afford to neglect getting good writers to produce good texts, and that the production of such texts should be a major goal of any on-line help project. In short, when you move documentation from paper to screens, very little changes: form, though important, remains clearly secondary to content.

Acknowledgements

Jim Morris, Dick Hayes, Phil Hayes, Kamila Robertson, and Frank Wimberly advised and helped me throughout this research. I am deeply grateful to them, and to the more complete list of people to be found in my dissertation. I also want to thank the General Electric Foundation and the National Science Foundation for their support during my graduate student career, and Advanced Programming Resources, of Columbus, Ohio, for their continuing support in the ongoing development of the ACRONYM help system.

References

[1] Borenstein, Nathaniel S. *The Design and Evaluation of On-line Help Systems*. PhD thesis, Carnegie-Mellon University, 1985.

[2] Borenstein, Nathaniel. The Evaluation of Text Editors: A Critical Review of the Roberts and Moran Methodology Based on New Experiments. In *Procedings of CHI '85*. 1985.

[3] Bramwell, Bob. BROWSE: An On-line Manual and System Without an Acronym. *SIGDOC Newsletter* , 1984.

[4] Draper, Stephen W. *The Nature of Expertise in UNIX*. HMI Project, University of California at San Diego, March, 1984.

[5] Fenchel, Robert S. *Integral Help for Interactive Systems*. PhD thesis, UCLA, 1980.

[6] Finin, Timothy W. Providing Help and Advice in Task-Oriented Systems. In *IJCAI 83 Proceedings*, pages 176-178. 1983.

[7] Fischer, Gerhard, Andreas Lemke, and Thomas Schwab. Knowledge-based Help Systems. In *Proceedings of CHI '85*, pages 161-167. 1985.

[8] Gosling, James. *UNIX Emacs Manual* 1983.

[9] Hayes, Philip J. Uniform Help Facilities for a Cooperative User Interface. In *National Computer Conference Proceedings*, pages 469-474. AFIPS, 1982.

[10] Houghton, Raymond C., Jr. Online Help Systems: A Conspectus. *CACM* 27(2):126-133, February, 1984.

[11] Magers, Celeste S. An Experimental Evaluation of On-line Help for Non-Programmers. In *CHI '83 Proceedings*, pages 277-281. 1983.

[12] Price, Lynne A. Thumb: An Interactive Tool for Accessing and Maintaining Text. *IEEE Transactions on Systems, Man, and Cybernetics* SMC-12(2):155-161, March/April, 1982.

[13] Relles, Nathan and Lynne A. Price. A User Interface for Online Assistance. In *Procedings of the Fifth Conference on Software Engineering*, pages 400-408. 1981.

[14] Relles, Nathan, and Norman K. Sondheimer. A Unified Apporach to Online Assistance. In *National Computer Conference Proceedings*, pages 383-387. AFIPS, 1981.

[15] Roberts, Teresa L. *Evaluation of Computer Text Editors*. PhD thesis, Stanford University, 1979.

[16] Roberts, Teresa L. and Thomas P. Moran. The Evaluation of Text Editors: Methodology and Empirical Results. *CACM* , April, 1983.

[17] Shneiderman, Ben. *Human Factors Issues of Manuals, Online Help, and Tutorials*. Technical Report CS-TR-1446, Department of Computer Science, University of Maryland, September, 1984.

[18] Sobell, Mark. *A Practical Guide to the UNIX System*. The Benjamin/Cummings Publishing Company, Menlo Park, California, 1984.

[19] Sondheimer, Norman K. and Nathan Relles. Human Factors and User Assistance in Interactive Computing Systems: An Introduction. *IEEE Transactions on Systems, Man, and Cybernetics* SMC-12(2):102-107, March-April, 1982.

[20] Wilensky, Robert. Talking to UNIX in English: An Overview of an On-line UNIX Consultant. *AI Magazine* 5(1):29-39, Spring, 1984.

Table 1: Summary of Tasks in the Experiments

Order	Intermediate task *(solution)*	Expert Task *(solution)*
T_1	Time of day *(date, uptime, whenis)*	Print on dover specifying font *(cz -f)*
T_2	Change password *(passwd)*	Sort in reverse order *(sort -r)*
T_3	List files *(ls)*	List files by time modified *(ls -t)*
T_4	View file *(cat, pr, more)*	Change protection as specified *(chmod o-r, chmod 640)*
T_5	Copy file *(cp)*	Delete file reversibly *(del)*
T_6	Rename file *(mv)*	List deleted files *(lsd)*
T_7	Print file on dover *(cz)*	Restore deleted file *(undel)*
T_8	Delete file reversibly *(del)*	Find i-number *(ls -i)*
T_9	List deleted files *(lsd)*	Set setuid bit *(chmod u+s, chmod 4xxx)*
T_{10}	Restore deleted file *(undel)*	Send to user logged in twice *(send -all, send user ttyxx)*
T_{11}	Direct message to another user *(send, write)*	List processes for all users *(ps a)*
T_{12}	Print calendar *(cal)*	Print file on dover with header *(cz -h "...")*
T_{13}	View file backwards *(rev)*	Sort, ignoring capitalization *(sort -f)*
T_{14}	Print working directory *(pwd)*	Cancel all pending mail requests *(mailq -retain)*
T_{15}	Make new directory *(mkdir)*	View only the printable strings in a binary file *(strings)*
T_{16}	Change directory *(cd, chdir)*	Execute remote command *(cmuftp r g - = "[date")*
T_{17}	Move file *(mv)*	Undelete old version of file *(undel -g)*
T_{18}	Delete empty directory *(rmdir, rm -r)*	Retrieve file from Onyx server *(ecp -u guest guest "[onyx]<AltoDocs>chat.tty" chat.tty)*
T_{19}	Delete full directory *(rm -r)*	Send to user on remote machine *(rsend user@host, send user@host)*
T_{20}	List current users *(u, users, finger, who, w)*	List processes on terminal ttypa *(ps tpa)*
T_{21}	Find string in file *(grep)*	List process with no terminal *(ps tp?)*
T_{22}	Send mail *(mail)*	List total space occupied by deleted files *(lsd -t)*

Table 2: Help Systems Studied in the Experiment

Help System	Description
H_0	Standard CMU UNIX help system (man/key)
H_1	Hybrid system: man/key with texts from ACRONYM
H_2	Fully implemented prototype system (ACRONYM)
H_3	Ever-present human tutor
H_4	Simulated natural language help system.

Figure 1: Sample view of ACRONYM's screen

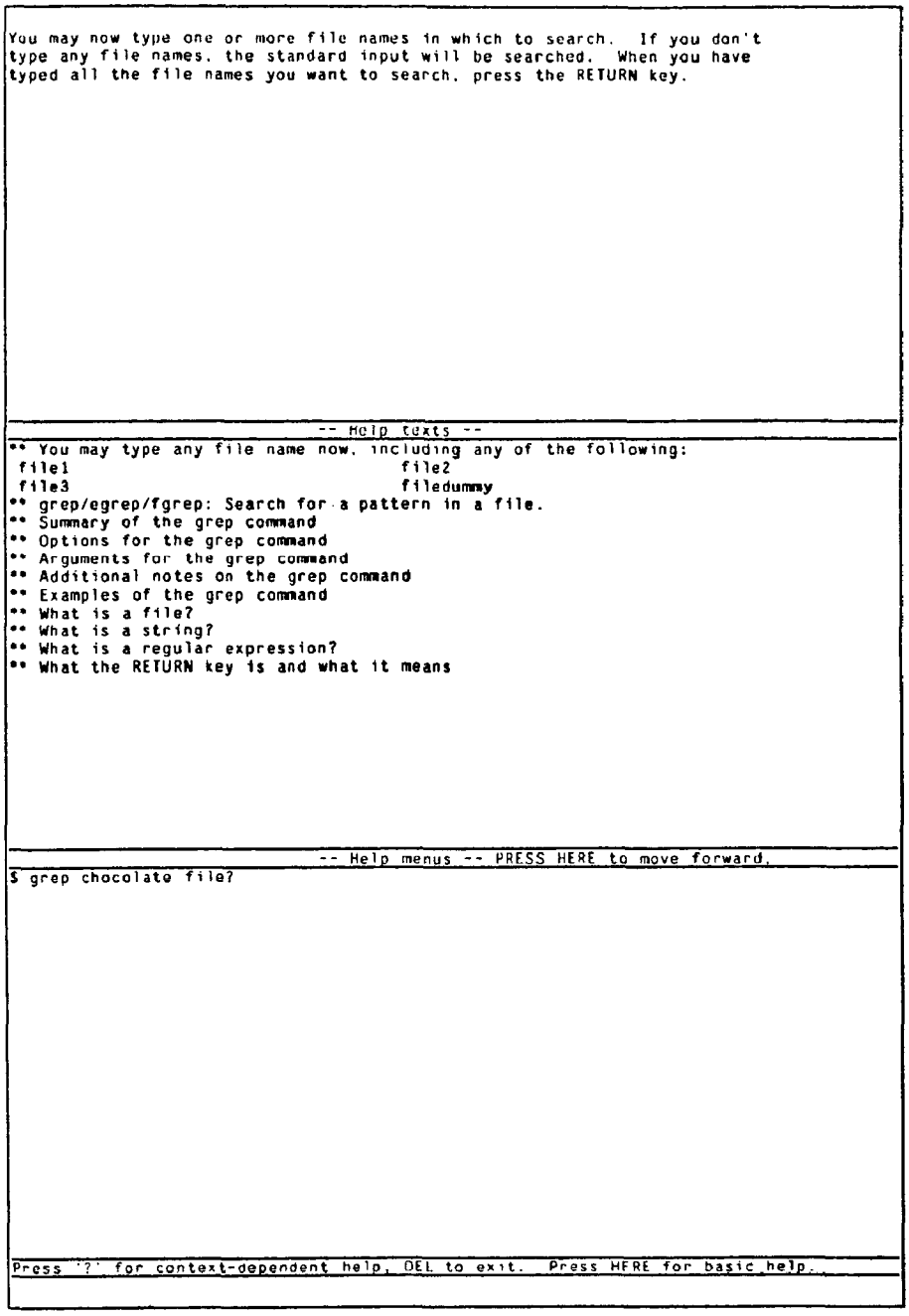

Figure 2: Distribution of Average Novice Timings

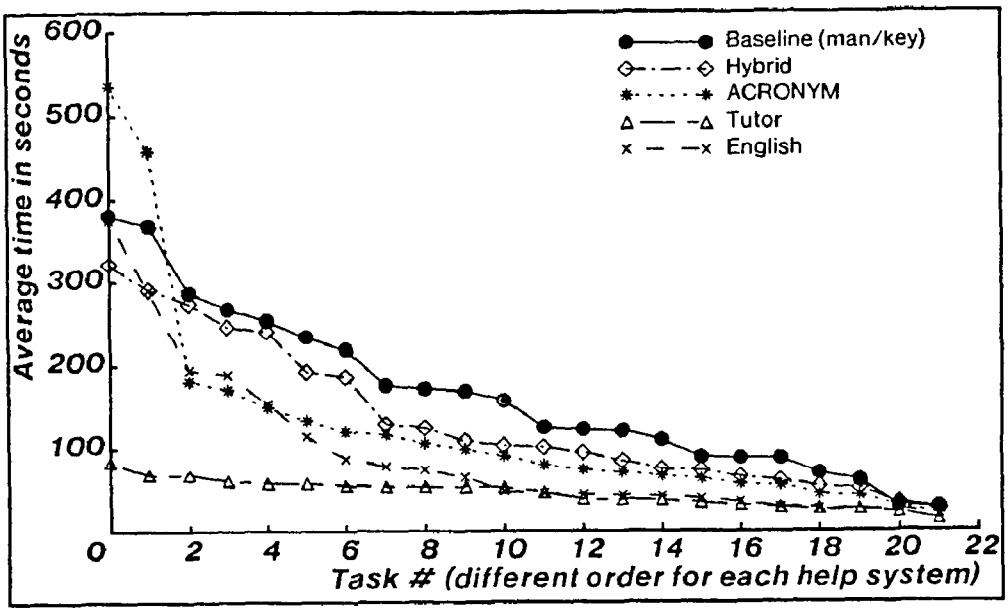

Figure 3: Distribution of Average Expert Timings