Playback: A method for evaluating the usability of software and its documentation

by A. S. Neal
R. M. Simons

Human factors evaluations of software products and accompanying user publications must be conducted so that developers can be certain that the target user population can learn to use the product with a minimum of difficulty and be able to perform the intended tasks efficiently. A methodology is described for obtaining objective measures of product usability by collecting performance data on the user interface without affecting the user or the system being evaluated. The log of stored activity is later played back through the host system for analysis.

Developers of programs and their documentation can no longer expect that their products will be used exclusively by data processing professionals. Therefore, these products must be easy for all intended users to learn and use. We are regularly reminded of this fact by product advertising in all popular communications. Attention must be focused on the personal interfaces to products and systems to ensure their usability. Human factors specialists are now assisting in the product development process. These specialists have the following roles in product development: (1) they provide advice and counsel on the user-interface design, (2) they interpret design guidelines, (3) they compare alternate designs, and (4) they conduct product evaluations.

Comparisons of alternate designs and product evaluations require that the products be tested. Human factors testing of hardware components of computer interfaces has been performed frequently, but the testing of software has been limited primarily because of the complexity of the task. Tests of the usability of software products and their explanatory publications need to be conducted to be certain that the target user population will be able to learn to use each product with minimum difficulty and apply them with the maximum efficiency that their developers intend.

More than a single test is usually required to accomplish these usability goals. For example, an initial test yields information that can be given to product development and publications groups for corrective action. After indicated changes to the product, a second test is made to determine whether those changes have alleviated the problems discovered in the first test and to detect additional problems that the changes may have caused. Frequently, this process is repeated several times.

The earlier in the development cycle that testing begins, the more efficiently product developers can make changes. Testing can begin long before the coding phase of the development cycle by employing prototypes of the user interface.

The purpose of usability testing is to determine whether a product is easy to learn and use, as well as to detect areas for improvement. Usability goals need to be specified in terms of ease-of-use criteria that can be measured. An objective test must consist of typical users actually working with a product. Observations and measurements of users' performance can be recorded during and after training. Users' comments, suggestions, and preferences should be solicited only after they have had some practical experience with a product.

Usability evaluation. In usability testing, the test objectives and scope must first be defined, and a decision as to what parts or aspects of a product to test should be included. For example, the test might be designed to evaluate the user interface of an entire product, or the focus might be on such individual features as the following:

- Command syntax.
- Task procedures.
- Screen layout and context.
- Menu navigation.
- Messages.
- On-line help facility.
- Publications.
- Training.
- System response time.

Operating software is needed for those parts of a user interface to be tested. If the testing is to be done before actual code is available, a prototype of the user interface can be created. Prototypes can range in complexity from simple straight-line simulations to functioning interfaces. Bury[1] used a straight-line simulation in a usability evaluation of a record-selection language for a word-processing machine. In that evaluation, if the query statement was correct, a canned correct result was displayed. To simplify the simulation, an incorrect query result was not presented; instead, only a message stating that the query was incorrect was presented. In the same report, Bury also discusses using an elaborate prototype of a user interface to a programming language, where most of the product's edit and display functions actually worked. Prototypes are also useful in comparing design alternatives, because the construction of a prototype is much easier and less expensive than producing multiple versions of an actual product.

Test software must run on a system that provides at least as short a system response time as that expected with the final product in a field environment. Excessive delays may adversely affect the user's learning and the operation of the product. The terminal hardware also should be typical of the devices expected to be used with the software in the field.

It is important to provide the subjects with motivation approximating that provided by a potential employer.

The user documentation for a product can and usually should be tested at the same time as the software. Current drafts of product tutorials, programmers' guides, and reference books should be obtained. Decisions must be made concerning which books or parts of the books should be used and evaluated.

The target user population for which a product is intended must be determined; product specifications are often vague about intended users. Representative test subjects are selected from the population of intended users. If test subjects from this population are difficult to obtain, surrogate users are identified and agreed upon by all parties involved. Even when test subjects are obtained from an appropriate user population, it is important to provide the subjects with motivation that approximates that provided by a potential employer.

Another essential ingredient for a usability test is the selection of tasks for the users to perform with the product. Test goals determine the types of tasks selected. If the objective is to measure the effectiveness of a training program or tutorial, one of the tasks is the training itself, including required exercises. Subsequent to training, the users should be asked to perform a selection of tasks designed to measure how well they have learned to use the various features of the product. Other tasks can be employed to assess usability and productivity. These tasks should be representative of the types of activities the ultimate users of the product will perform.

Usability testing must initially be conducted in a controlled environment. To obtain valid and reliable usability measurements, the tester must know how the user was trained, what portions of the system were available, and what testing tasks the user performed. Later, field studies can be run to identify special problems associated with the integration of the product into an actual working environment. Erdmann and Neal[2] discuss further the relative purposes and merits of controlled laboratory and field studies.

Comparative measurements of usability can be made on alternate product designs.

Finally, systematic measurements must be made of user performance in order to obtain the quantitative and qualitative information needed to determine the level of usability of the product. Measurements of user performance are useful for comparing a current product with earlier versions of that product to see whether the revisions have yielded improved performance. Comparative measurements of usability can be made on alternate product designs. Human factors specialists can analyze, summarize, and interpret these measurements in order to recommend changes in both the software and the documentation to optimize product usability.

Measurements to be collected are determined by the goals of the test and the ease-of-use criteria that may have been established for the product. Measurements of ease of learning may include the following:

- Time required to complete a training program.
- Total time needed to achieve a given performance criterion.
- Observed difficulties in learning the product.
- User comments, suggestions, and preferences collected in a postlearning interview.

Objective indicators of user difficulties could be the following:

- Frequency with which each error message is encountered.
- Inability to find needed information in the documentation.
- Frequency of use of each section of an on-line help facility and the amount of time spent using help.
- Number of times the user asks for special assistance (perhaps by calling a simulated product support center).
- Efficiency with which the user employs different features.

Measurements of ease of use after initial learning may include the following:

- Time required to perform selected tasks.
- Success or failure in completing tasks.
- Frequency of use of various commands or language features.
- Time spent looking for information in documentation.
- Measures of user problems similar to those used to measure learning difficulties.
- User comments, suggestions, and preferences.

In this paper, we discuss a methodology we call *Playback* that we have developed for testing program products and their documentation. We first discuss this methodology in general and relate it to the general principles just presented. We then discuss Playback analysis, data collection and recording, program management, and our experience with Playback.

Playback methodology

The experimental methodology called Playback has been developed at the IBM Human Factors Center to make the kinds of measurements just described, in order to evaluate the usability of software and/or software documentation. Playback evolved over several years and has been employed at the Human Factors Center to evaluate a variety of software packages operating on systems ranging from standalone word processors to large multipurpose computers. Several of these studies are presented in the last section of this paper.

The central idea of Playback is that, while a user is working with the system, the keyboard activity is timed and recorded by a second computer. This stored log of activity is later played back through the host system for observation and analysis. Thus, the

methodology is noninvasive in that the data-collection programs are external to the product being evaluated, and the method is nonintrusive because the data collection does not infringe upon the user's activities.

Basic system operation

Figure 1 is a block diagram of the basic testing system. The user sits in front of a terminal that is attached to a host system. The user's keyboard is not connected directly to the display terminal or to the host. Instead, the keyboard is connected through interface logic to the Human Factors Center laboratory computer.

The laboratory computer time-stamps and records each keystroke. The time recorded is the cumulative time in milliseconds since the start of the testing session. The laboratory computer then transmits the keystrokes to the keyboard interface of the host system terminal. All the data-collection programs run on the laboratory computer. In contrast, if data-collection programs were to reside in the host system, details of all keying activity, including accurate timing information, might be lost because some computer terminals send nothing to the host until certain interrupt keys are used. With data collection being conducted in the laboratory computer, every keystroke can be timed and recorded.

No modifications are necessary to the host computer software that is being evaluated. The host software may be actual product code or a prototype of the user interface to a software product under development.

Session initialization. The auxiliary display shown in Figure 1 is used to initialize the test session, to present task instructions, and to display the Playback analysis. The usability evaluation is divided into separate testing sessions. A session can consist of reading and doing exercises in one chapter of a tutorial, or it can include the execution of one task of a performance test. Before beginning a test session, the experimenter usually sets up the host system to the state at which the user begins execution of a particular task that involves the software under evaluation. In addition, the experimenter can enter information that identifies the subject, task, and test conditions.

Whether reading and doing exercises or executing a performance test, both objectives are accomplished

Figure 1 Configuration of the Playback system

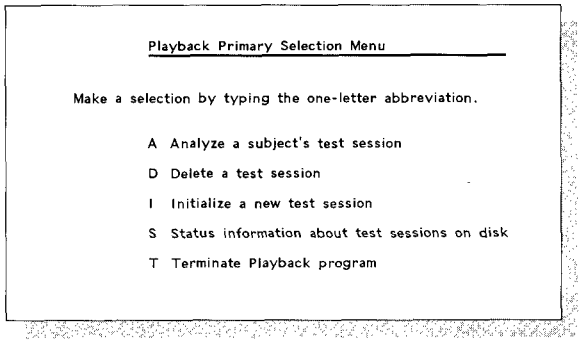
by software switches in the Playback program on the laboratory computer. A unique combination of switches controls the connection to the host system. One switch determines whether the keystrokes will be transmitted to the host. Another switch determines whether keystrokes will be interpreted and recorded by the laboratory computer. The status of both switches is shown on the auxiliary display.

When setting up the host system for a session, the experimenter uses these switches to turn the host system on and the laboratory system off. The laboratory system automatically uses the opposite switch positions (host off, laboratory on) when accepting identifying information about the session. Both host and laboratory switches are turned to *on* just before the user begins to work.

Session identification information is provided to the laboratory computer by selecting I (Initialize a new test session) from the primary Playback menu shown in Figure 2. The program then prompts the experimenter for a subject (user) number, session number, and condition number. In subsequent sessions, the previously entered subject and condition numbers will be displayed for verification, and the session number will be incremented by one. These numbers are displayed one at a time and are verified by the experimenter by keying the Line Return key. Corrections can be made by backspacing and rekeying.

In the final step in this initialization process, the Playback program requests the experimenter to verify all the identifying numbers. If the experimenter indicates that the numbers are incorrect, the program allows the experimenter to step through the numbers again to make corrections.

Figure 2 Primary Playback menu



These session identifying numbers are retained with the data collected during the session. The condition number identifies experimental conditions such as order of presentation of tasks or alternate product designs being tested.

Task description. After the experimenter verifies the identifying numbers, the message WHEN READY, PRESS SPACE BAR is displayed on the auxiliary screen by the program. After ensuring that the appropriate documentation is available to the user and that the host system is ready, the experimenter leaves the room. When the user presses the space bar, the Playback program displays on the auxiliary screen a description of the required task to be performed in this session. The description can be as simple as "Read Chapter 3 of the tutorial and do the prescribed exercises using the ABC system," or as extensive as a complete description of a computer program to be written on the host system.

Keystroke monitoring. The user now attempts to accomplish the task, using the host system software and appropriate documentation. All keystrokes made by the user are time-stamped and recorded for later analysis. In addition, the Playback program sends to the host system a predetermined interrupt if no user interrupts have occurred during a selected time interval. This is to prevent automatic logoff on some systems.

Documentation monitoring. We are sometimes interested in evaluating not only the user's interactions with the software but also with the documentation. We want to know how easily the user can find the information needed to solve a problem. In these evaluations, an observer in a separate room monitors the use of the documentation or book activity via television. Figure 3 shows a typical observation station arrangement. One television monitor provides an overview of the work station, giving the observer a view of the user, display, keyboard, and documentation. This monitor is used to determine whether the user is looking at the terminal or the documentation.

A second monitor shows a close-up of the documentation or books available to the user. Large page numbers are written in the books so that they are readable on the television monitor. Although the printed matter is not generally readable on the monitors, the observer has a copy of the books in the observation room for reference. The user is required to keep the book or books within certain confines of a table so that they will be within the range of the camera. Alternatively, a book box that contains a platform for the book and a mount for lights and camera is sometimes employed. The book box permits the user to move the book around for comfortable reading without disturbing the relative positions of camera and book.

The observer may enter observation codes and comments about the user's activities.

In addition to the overview monitor and the book monitor, the observer sometimes employs a slave display that shows the identical information that is on the user's host system display. The observation station is also equipped with a keyboard-display terminal connected to the laboratory computer. An example of this display is shown in Figure 4. The subject, session, and condition numbers established during session initialization are shown at the top. At the bottom of the screen is a readout that shows the cumulative session time.

In the center of the display appear three columns, labeled TIME, CODE, and COMMENT. The observer may enter observation codes and comments about the user's activities. These entries are time-stamped and recorded by the laboratory computer. The time re-

corded is the cumulative session time—the same time indicator recorded with the keyboard activity.

Five-character alphanumeric codes can be used. The experimenter selects easy-to-remember codes to denote such specific book activities as, for example, using the Index, browsing in Chapter 3, reading page 42, or looking at page 19 while keying.

The cursor on the observer's display is initially located in the code field. When the first keystroke of a code is entered, the current session time is displayed in the time field. Corrections in an entry can be made with the back-space key. End of the code is signaled by pressing ENTER or TAB. If TAB is pressed, the cursor moves to the comment field to allow the experimenter to enter a comment of any length. If the entry is too long to be displayed on that line, the cursor is automatically moved to the next line in the comment field. Alternately, the observer can move the cursor to that location using the line-return key. An enter key signals completion of that comment.

The observer station is usually manned by the experimenter during pilot testing to debug procedures and tasks and to establish an appropriate set of codes. Later the observer station may be manned by a laboratory assistant who is not required to understand the host system operation or details about the tasks being performed.

Requests for assistance. A function key is provided the user to request assistance from the experimenter. This assistance is separate from and in addition to any on-line help function that might be available in the host software. The user might request assistance for various reasons including lack of understanding of task instructions, inability to solve the problem, or malfunction of the host system. The Playback program acknowledges requests with an audio signal in the testing room, a signal light outside the testing room, and messages written on the user's auxiliary screen and the observer's display. This assistance condition is turned off and the user resumes work upon a second depression of the Assistance key.

If an observation station is employed in the evaluation, information about the nature of the assistance request and the type of assistance given is recorded via codes and comments entered on the observer's terminal.

Session completion. The user presses a function key labeled DONE to indicate that the task is completed satisfactorily. The Playback program acknowledges the done key in a manner similar to its response to a request for assistance. The session timer is stopped immediately, but the observer is allowed to complete any code or comment entries already started. Finally, the observer's display and the user's auxiliary display are erased and the summary data for this session are stored.

Stored with the task description for a session is a code indicating whether this is a *terminal* or *continuous* session. If it is a continuous session, the subject, session, and condition numbers are displayed for the next session, along with the message to press the space bar when ready. The observer's display is also updated for the next session. The user may take a break at this point because the session timer is not started until the space bar is pressed. Continuous sessions can be employed only if no setup of the host system is required between sessions.

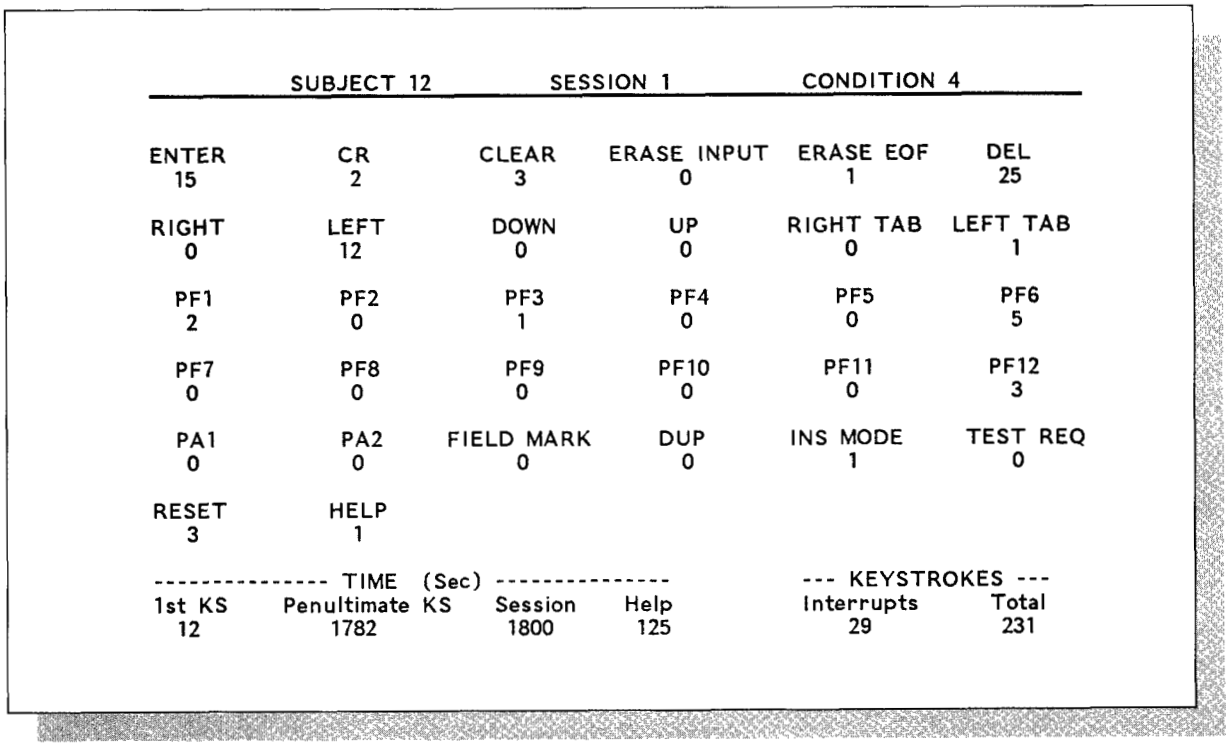
Figure 3 The observer station

Figure 4 Example observer display

Subject 12 Session 1 Condition 4

TIME	CODE	COMMENT
0:03:59	P42	*Studying Programmers Guide. Seems confused!*
0:04:15	M1	Subject requested a coffee break
0:18:51	M2	
0:18:57	P83	
0:20:38	P19K	
0:22:02	RTC	*Searching for topic in Table of Contents*
0:22:46	R178	
		0:23:03

Figure 5 Example statistics display



SUBJECT 12 SESSION 1 CONDITION 4

ENTER	CR	CLEAR	ERASE INPUT	ERASE EOF	DEL
15	2	3	0	1	25
RIGHT	LEFT	DOWN	UP	RIGHT TAB	LEFT TAB
0	12	0	0	0	1
PF1	PF2	PF3	PF4	PF5	PF6
2	0	1	0	0	5
PF7	PF8	PF9	PF10	PF11	PF12
0	0	0	0	0	3
PA1	PA2	FIELD MARK	DUP	INS MODE	TEST REQ
0	0	0	0	1	0
RESET	HELP				
3	1				

TIME (Sec)

1st KS	Penultimate KS	Session	Help
12	1782	1800	125

KEYSTROKES

Interrupts	Total
29	231

If DONE is pressed for a terminal session, a message on the user's auxiliary screen asks him to wait for the experimenter. At this point, the experimenter may request the Playback program to present summary statistics on the user's auxiliary screen. Figure 5 shows a sample of such a display. Shown are the subject, session, and condition identifiers; frequency counts of functions, commands, and keystrokes; and various timing measures. The experimenter may desire to discuss some of these measures with the user for motivational or tutorial purposes.

During this break between sessions, the experimenter has the opportunity to make any setups on the host system that may be necessary for the next session. The experimenter has the capability to abort a session and readjust the session identifiers, in the event of operational difficulties.

Playback analysis

Playback analysis is the key feature of the Playback program. Each session is separately stored in the laboratory computer. The experimenter usually conducts a Playback analysis of the user's performance after the user has completed all required sessions. There are occasions, however, when the playback is done immediately after a session with the user present, in order to obtain supplementary information about the user's thoughts or reasons for particular actions while performing the task. Playback analysis can be performed many times if necessary.

Playback setup. To conduct a playback analysis, the experimenter selects A (Analyze a subject's test session) from the primary Playback program menu shown in Figure 2. The Playback program running on the laboratory computer asks the experimenter for the subject (user) and session numbers to be analyzed. The experimenter then uses the special switches, discussed earlier, to turn off the laboratory computer and turn on the host system, in order to set up the host program to the state that existed prior to the time the user began the session to be analyzed. The special switches are then set so that only the laboratory computer is on.

Playback pacing. The experimenter then requests Playback to pace through the user's actions during the session. This is accomplished by pressing one of four function keys to cause the laboratory computer to send one of the following sequences of user keystrokes to the host system:

- Next keystroke.
- All characters keyed up to and including the next function selected.
- All characters and functions up to and including the next interrupt.
- All characters up to and including the next function with the same time intervals as when originally keyed by the user.

Each time the experimenter presses one of these function keys, the laboratory computer sends the appropriate characters to the host system so that the host terminal display appears just as it appeared to the user at that same point in the test process. The experimenter may switch between pacing methods at any time during the Playback analysis.

Playback display. At the same time the laboratory computer transmits the characters to the host system it also writes the characters on an auxiliary display. An example of the display the experimenter might see while pacing through the user's actions is shown in Figure 6. The previous function or interrupt the user selected is displayed along with the cumulative session time at the point the function was keyed. All the keystrokes up to the next function or interrupt are shown on the following lines, along with the session time when the first keystroke was made. The next line shows the subsequent function or interrupt. The display also shows the time interval between entering the two functions. Also shown on the display are the observer's codes and comments that were entered from the observation station during this period, along with the session time when they were entered. The next observer code and comment subsequent to the current time are also shown.

Occasionally, when pacing through a session, the experimenter desires to back up and replay a short segment of the action. A back-up key is provided for this purpose. When the back-up function is used, the Playback display shows the previous interrupt. One can back up all the way to the beginning of the session if that is desired. Because it is not feasible to back up the host system, the host display remains

Figure 6 Sample Playback screen

```
Subject 12          Session 1          Condition 4

0:20:30   CLEAR
0:20:55   A = (22/7) * R**2
0:21:20   ENTER                      Function interval: 50 sec.

------------------------------  OBSERVATIONS  --------------------------

0:20:38   P19K
0:22:02   RTC   Searching for topic in Table of Contents




--------------------------------------------------------------------------
PF1: Next keystroke   PF2: Next function   PF3: Next interrupt
PF4: Real time        PF6: Set time        <--: Backup      CLEAR: Abort
--------------------------------------------------------------------------
0:21:20   E5      Syntax error because referring to wrong page in
                  Programmers Guide
```

static until forward pacing is resumed and catches up to the point where the backing-up began.

Problem determination and recording. As the experimenter paces through the user's activity, he can enter additional codes and comments into the data stream in the same way they were entered by the observer during the session. These codes and comments may be additional observations not noted originally, or they may be indicators of certain errors or problems noted during analysis. These codes are also time-stamped with the cumulative session time, so that in later review there is a record of the point in the session at which errors or problems occurred.

A detailed analysis requires the experimenter to be very familiar with the software, documentation, and user's task.

A detailed analysis of the user's actions during Playback requires the experimenter to be very familiar with the software and documentation being evaluated, as well as the nature of the user's task on that session.

Data collection and recording

Tape records. All user keystrokes and the cumulative session time in milliseconds are recorded on tape during the session. Also recorded on tape during the session are the experimenter's codes and comments entered from the observation station (along with the associated session time). In addition, the following statistics are collected during the session and written on tape at the conclusion of the session:

- Time from session beginning (space bar) until first user keystroke.
- Time from session beginning until last keystroke.
- Cumulative time in an "Assistance" condition.
- Total session time from session beginning to DONE key.
- Frequency of use of each function key.
- Number of requests for assistance.

Also recorded on tape are the experimenter codes and comments entered during Playback analysis. All data on tape for a session are preceded by an identification record containing the subject (user), session, and condition numbers. The tape records are for archival purposes, allowing later analysis of other information that was not required at the time of testing.

Disk records. In addition to the above tape records, a disk data set is written by the Playback program to include the user's keystrokes and the codes and comments (entered both during the session and during the playback analysis). This data set is used to accomplish the Playback analysis. It may also be displayed or printed by the experimenter. Several options are available. The display or printout may contain only the user's keyboard activity, only the codes and comments, or both. Figure 7 is a sample printout of this data set, showing both types of entries. The first column is the cumulative session time in hours, minutes, and seconds. The second column shows the codes entered by the observer during the session and codes entered by the experimenter during Playback analysis. The third column contains either the user's keystrokes or the observer/experimenter comments. If user keystrokes are shown in the third column, the time field shows the session time for the first keystroke, and the code field remains blank. Function keys are listed on a separate line. Continuation lines are allowed for the comments, and these are indicated by a quotation mark (") in the code column.

Data analysis. The Playback program also allows the experimenter to perform limited analysis of this data set to obtain measurements such as the following:

- Frequency of selected experimenter-entered codes.
- Time between selected pairs of codes.
- Time between selected code and next user keystroke.

This information may be accumulated over a single session or over many sessions. If needed, more sophisticated analysis of the user- or experimenter-entered data may be performed using *ad hoc* programs for manipulating either the disk or tape records.

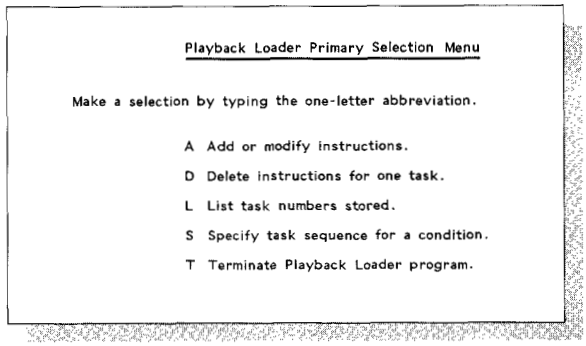
Figure 7 Printed record of a portion of a session

```
SUBJECT  40        SESSION  1        CONDITION  40

 TIME      CODE   SUBJECT KEYSTROKES / OBSERVER COMMENTS

0:03:59    P42    Studying Programmers Guide
           "      Seems confused!
0:04:15    M1     Subject requested a coffee break
0:18:51    M2
0:18:57    P83
0:20:30           <CLEAR>
0:20:38    P19K
0:20:55           A = (22/7) * R**2
0:21:20           <ENTER>
0:21:20   *E5     Syntax error because referring to wrong page in
           "      Programmers Guide
0:22:02    RTC    Searching for topic in Table of Contents
0:22:38           <CLEAR>
0:22:46    R178
0:23:30    E
0:25:00           )PCOPY WS
0:25:14    H      ASKED ABOUT LOADING FROM PUBLIC LIBRARY
           "      AFTER EXPLAINING, SHE KNEW TO CLEAR WS
           "      AND USE PCOPY
0:25:59           )PCOPY CREDIT GETANS
0:26:22           <ENTER>
0:26:29           )PCOPY CREDIT GETINFO
0:26:42           <ENTER>
0:26:51           )FNS
0:26:53           <ENTER>
0:27:02    B
0:27:03    54
0:27:13    E
0:27:19           WS2 C2
0:27:33           <ENTER>
0:27:51    B
0:27:57           )
0:27:59    E
0:28:00           PCOPY 3 WS2 'C2'
0:28:11           <ENTER>
0:28:36    B
0:28:37    56
0:28:43    57
0:28:58           <PF8>
0:29:01           <CLEAR>
0:29:04    E
```

Program management

Concurrent experiments. Multiple copies of the Playback program may be loaded in the same Human Factors Center computer. Each copy controls one user and optionally one observer station. All experimental data are recorded on a community tape that is later separated by user.[3]

Figure 8 Primary menu for the Loader Program



Job parameters. Before the Playback program can be executed, multiple copies of the program must be created, several data sets must be created, and data definitions must be established for each data set used by Playback. In order not to burden the experimenter with programming details, an interactive EXEC that performs all the preceding functions is provided. The EXEC interrogates the experimenter for the following parameters of his experiment:

- Name of experiment.
- Number of subjects' data to be stored concurrently.
- Maximum number of keystrokes per subject.
- Maximum number of sessions per subject.
- Maximum number of observations per subject.
- Average length of task instructions.

When the Playback program is initially loaded, such job parameters as the following are specified:

- Input and output computer ports for the various keyboards and displays.
- Whether an optional observation station is used.
- Whether separate task descriptions are to be used or a common description as a default.
- Default maximum session time.
- Whether end-of-session statistics should be displayed.

These and other job parameters are saved. Thus, when Playback is later executed, only those parameters that the experimenter wants to change need be entered. Ordinarily all parameters remain unchanged from run to run.

File management. The selection of S (Status information about test sessions on disk) on the Playback program's primary menu shown in Figure 3 allows the experimenter to note the disk status by displaying the number of sessions recorded on disk for each subject number. Also shown is the percentage of allocated disk space that is filled. The selection of D (Delete a test session) allows the experimenter to delete sessions no longer needed on disk.

Playback loader. A separate interactive program, which also runs on the same Human Factors Center computer, performs several functions necessary for preparing an experiment using the Playback program. The primary display menu for this Playback Loader program is shown in Figure 8.

The selection of item A (Add or modify instructions) on the Playback Loader menu allows the experimenter to enter or modify the task descriptions presented to the test user at the beginning of each session. These task descriptions are entered with an arbitrary identifying number. Figure 9 shows a sample display from this function of the Playback Loader program. In addition to the entry of the task description, this function requests a maximum time to be allowed for the task (0 specifies no limit). Also, a specification can be entered to indicate whether, at the conclusion of the task, the experimenter should be called (i.e., a *terminal session*) or whether the next task should be presented (i.e., a *continuous session*).

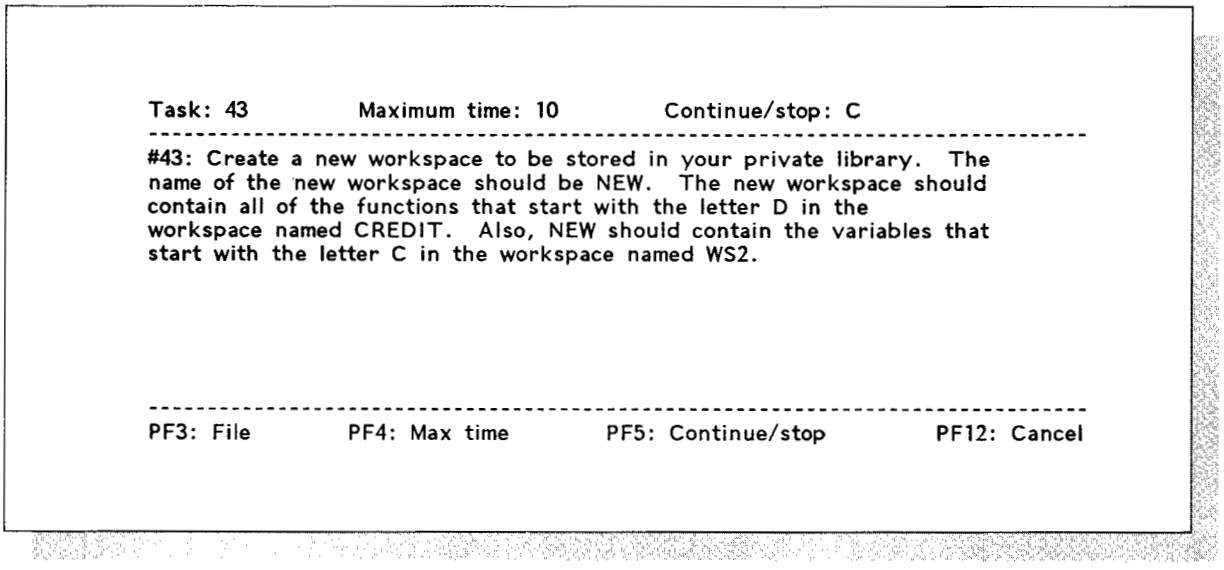
Selection of item D on the menu (Delete instructions for a task) allows the experimenter to delete one of the previously entered task instructions. Selection of item S on the Playback Loader menu (Specify task sequence for a condition) allows the experimenter to specify the order of presentation of the tasks for each test condition. Function keys are used to insert and delete tasks from this list. Selection of item L (List condition numbers stored) lists the condition numbers for which task indexes have been entered.

It is not necessary to use Playback Loader before using the Playback program if defaults are preferred. The default task instruction is the following: COMPLETE THE TASK FOR SESSION ____. In the default, all test users receive the tasks in the same order, all sessions are terminal sessions (experimenter is called), and the maximum session time is the same for all sessions.

History and applications

The first application of the Playback process at the Human Factors Center was in a study of word-

Figure 9 Sample task description



processing machine functions and displays conducted in 1980. Users' keyboard activity, which had been collected earlier while they performed selected editing tasks, was replayed through a simulated word processor in order to tutor the users. In this study, the programs simulating the host system and the programs for data collection and for playing back tasks all ran on one of the IBM System/7 computers in the Human Factors Center.

Clauer[4] recognized the usefulness of the playing back of tasks as a method of data collection. He adapted this technique in an evaluation of the self-training material for a free-standing word-processing machine. The host system was an IBM Displaywriter. An early version of the Playback program ran on an IBM System/7 computer. Keystrokes were collected and recorded while users performed training exercises and did test problems with the Displaywriter. The users' keystrokes were later replayed through the Displaywriter in order to observe problems in the use of either the systems or training book. These problems were recorded by the experimenter during the paced replay by entering error codes into the Playback program to designate the section of the training book where problems occurred. Experimenter comments were also recorded.

The number and severity of user problems were the primary data item collected, along with training time, test completion time, and frequency of calls for assistance. Four interactions of this test were conducted, with modifications being made in the training material as a result of the findings of each interaction of the test. The process resulted in very worthwhile improvements in the training, as reflected in several of the performance measures. There were fewer requirements for assistance, fewer significant difficulties, shorter criterion test times (reflecting greater retention), and also somewhat higher subjective ratings of the training manual and skill proficiency.

The Displaywriter test clearly proved the utility of the Playback approach for the evaluation of the effectiveness of training material for a small standalone system. The next application of the technique at the Human Factors Center was in a study of how best to introduce novice users to the concepts of an interactive data base query language. The host system was a prototype of a query language running on an IBM System/370 VM system.

Additional Playback pacing methods were incorporated into the Playback program before its next use in a human factors study of IBM BASIC by Bury.[1] The study included an evaluation of a tutorial, an editor, and other interactive aspects of the language. The host system was, at first, a prototype running on a VM system. The user learned how to use the language

and editor from a tutorial that required exercises to be entered on an IBM 3277 Display Station. After completion of training, users had to write a number of programs using IBM BASIC. User actions were recorded and later played back through the host system to observe user problems with both the software and the training materials.

Six iterations of this study were conducted, with improvements in both the system and tutorial being made after each test. The first iterations were conducted using a prototype of IBM BASIC, while the later tests were conducted with the actual product code. No modifications were required in the Playback program running on the laboratory computer to accommodate this change. In almost all cases, improvements in user performance measures were achieved with new versions of the system and tutorial.

Experiences with Playback convinced us of the usefulness of the methodology to evaluate both training material and the software.

These experiences with Playback convinced us of the usefulness of the methodology to evaluate both training material and the software itself. The program was then completely rewritten to make it an efficient general-purpose laboratory tool. This new generalized version of Playback was first used in an evaluation of the training material for QMF, a data base query facility. Later, Ogden and Boyle[5] used the same program to compare three different methods of report modification after completing a query of a data base. Prototypes of the user interface for the three methods ran on a VM system. The Playback program was employed to conduct a replay analysis of the users' activities and to collect performance data. The performance data that were obtained clearly revealed the relative usability of the alternate designs. The product developers were then able to select for the product that design which yielded the best user performance.

This version of the Playback program was also used to study the usability of the advanced functions available in IBM BASIC. In this study, experienced programmers attempted to write BASIC programs to solve selected problems. They were required to use one or more of the advanced features on each task. The users needed the Language Reference Manual to learn how to use the advanced features, since none of the programmers were familiar with them. Although the emphasis in this study was on the advanced features of the software, inadequacies in the reference material were also revealed.

Late in 1982, several evaluations were planned for new versions of programming languages in which the emphasis was on the software documentation. Modifications in the Playback program again appeared to be called for. Since these modifications were substantial, a whole new Playback program was designed.

Coding of this new version was completed in the first part of 1983. The largest addition to the program was the capability of recording and displaying codes and comments entered from an observer station. The method of entering error codes and comments during Playback analysis was also modified, as was the method of storing the user's keyboard activity. Another major addition was the ability to display task instructions for each session. The Playback Loader program was developed to facilitate the entry of these instructions as well as the entry of specifications for such other new options as continuous sessions, varying task orders, and flexible session time limits. A number of other changes were made in the program to increase its utility and ease of operation.

The current version has now been used to evaluate the *Application Programmers Guide and Language Reference Manual* for a new version of a programming language. It has also been employed to test the effectiveness of a primer for novice users of an interactive system under development.

The early versions of Playback were all programmed to run on the Human Factors Center's IBM System/7 computers. The latest version has been converted to run on the Human Factors Center's IBM Series/1 computer network.

Although we feel that Playback is now a very flexible and useful human factors data-collection tool, we anticipate further improvements to be made as we

gain more experience using it in the evaluation of software and software documentation.

Concluding remarks

The Playback methodology has proved itself to be a very effective tool for objective evaluations and comparisons of software, including user interface design and software documentation. The Playback program incorporating this methodology has a number of attributes that make it flexible and useful.

Playback is a general-purpose data-collection program. No changes in the program are required from one application to another. Even radically different host systems (ranging from free-standing word processors to large multiple-user systems) can be monitored with little or no modification of the Playback program. Some setup, of course, is required for each individual experiment. Such preparations as entering task descriptions and various program options are handled by the experimenter using interactive features of the Playback and Playback Loader programs, without requiring the services of the programmer. The discussions in the previous section of the modifications made to the Playback program may have implied the contrary, but these changes were enhancements to the general utility of the program, and were not added specifically for any particular experiment.

The Playback methodology requires no modification of the host system software. All experimental control and data-collection functions are in the Playback program running on the laboratory computer.

The data-collection process does not intrude on the user's thoughts or activities. The user operates the actual or prototype system in a separate room without necessarily being aware of the extent of the data being collected on his activities. The experimenter, of course, tells the test user that his performance is being monitored. The user, however, perceives no time delay due to the Playback program capture of keystrokes before transmission to the host system, nor is he aware of any other interference.

The Playback program provides the unique capability for replaying the user's keystrokes back through the host system for later analysis. This allows the experimenter not only to know what keystrokes were made while the user interacted with the host system, but also allows the experimenter to see on the display what the user saw at each point in the execution of the task. The speed of Playback is controlled by the experimenter. Where the user had no apparent problem, the analysis can proceed quickly. Where there were problems, the experimenter can pace the review more slowly or even back up and replay sections requiring careful analysis. Our experience has been that an eight-hour day of user activity can usually be analyzed in an hour or two, depending upon the number of user problems, task complexity, and detail of analysis desired. This is certainly much less analysis time than would be required to review an equivalent amount of user activity had it been recorded on video tape only.

An eight-hour day of user activity can usually be analyzed in an hour or two.

Although the use of an observer station can be somewhat labor-intensive, the station can be manned by a laboratory assistant, thereby freeing the experimenter for other work. This is possible because problem determination is done during the playback analysis, rather than during the session.

Finally, all user actions and observer entries are recorded on-line. Data are thus immediately available for computer-aided summary and analysis.

Acknowledgments

The authors would like to express their appreciation for the valuable contributions made toward the development of the Playback methodology by a number of members of the Human Factors Center. Calvin Clauer was the first to apply the technique as a data-collection tool. James Boyle, Kevin Bury, William Ogden, and Barbara Isa used the technique in early stages of development and suggested improvements. Michael Darnell and Susan Wolfe debugged both the latest System/7 and Series/1 versions of Playback. They also used the program in several studies and helped improve the program's utility as

a general-purpose laboratory tool. The operation of Playback would not be possible without the contribution of Rob Cotton, who designed and built the interface logic that allows Playback to be used with a wide variety of stand-alone and terminal devices. In addition, the authors acknowledge ideas obtained from the IBM Product Usability group in Atlanta, Georgia, on techniques for monitoring the use of documentation.

Cited references

1. K. F. Bury, *Prototyping on CMS: Using Prototypes to Conduct Human Factors Tests of Software During Development*, IBM Human Factors Center Technical Report HFC-43, IBM General Products Division, 5600 Cottle Road, San Jose, CA 95143 (February 1983).
2. R. L. Erdmann and A. S. Neal, "Laboratory versus field experimentation in human factors," *Human Factors* **13,** No. 6, 521–531 (1971).
3. R. M. Simons, *A Community Tape*, IBM Human Factors Center Technical Report HFC-27, IBM General Products Division, 5600 Cottle Road, San Jose, CA 95143 (December 1977).
4. C. K. Clauer, "Methodology for testing and improving operator publications," *Proceedings of Office Automation Conference*, American Federation of Information Processing Societies, San Francisco, CA (1982), pp. 867–873.
5. W. C. Ogden and J. M. Boyle, "Evaluating human-computer dialog styles: Command versus form/fill-in for report modification," *Proceedings of The Human Factors Society 26th Annual Meeting*, Santa Monica, CA (1982), pp. 542–545.

Alan S. Neal *IBM General Products Division, 5600 Cottle Road, San Jose, California 95193.* Mr. Neal is the manager of the IBM Human Factors Center in San Jose, California. In the twenty years he has been with IBM, Mr. Neal has concentrated on designing and conducting experiments with the aim of optimizing the interface between computer systems and their users. Mr. Neal began his career with IBM in 1964 when he joined the Advanced Systems Division, where he worked as an engineering psychologist. In 1970 he transferred to the Research Division. He was a charter member of the Human Factors Center when it was formed in 1973, and was made a manager of interdivisional projects in the Human Factors Center in 1974. In 1981 he became manager of the software human factors group within the Human Factors Center. Mr. Neal was promoted to Senior Human Factors Engineer and manager of the Human Factors Center in August of 1982. He is a graduate of Purdue University (B.S.) and Iowa State University (M.S.) in experimental psychology. He has been an active member of the Human Factors Society since 1963, and is currently serving that organization as Publications Committee Chair and Managing Editor.

Roger M. Simons *IBM General Products Division, 5600 Cottle Road, San Jose, California 95193.* Mr. Simons is a Senior Programmer in the Human Factors Center in San Jose, California. In this assignment, he has developed programs that simulate the operator interface and measure operator performance for IBM products, including keyboards, displays, electronic typewriters, word processors, software, and software documentation. Mr. Simons joined IBM in 1955 at San Francisco, where he was an Applied Science Representative. In this position, he assisted IBM customers and IBM sales personnel in developing scientific computing applications. Since 1959, he has held various programming and management assignments in San Jose, including Manager of the Engineering and Scientific Computation Laboratory. Mr. Simons received a B.S. degree in mathematics from Stanford University and a Ph.D. degree in applied mathematics from MIT. He is a member of the Association for Computing Machinery and Sigma Xi, and is a former chairman of the Bay Area Chapter of the ACM.

Reprint Order No. G321-5211.