— METHODS & DESIGNS —

Heuristic and statistical support for protocol analysis with SHAPA Version 2.01

JEFFREY M. JAMES and PENELOPE M. SANDERSON
University of Illinois at Urbana-Champaign, Urbana, Illinois

This paper describes SHAPA Version 2.01, an interactive program for performing verbal protocol analysis. Verbal protocol analysis is a time-consuming activity that has hitherto typically been done by hand, whereas SHAPA represents an attempt to build a software environment to aid (but not replace) researchers in this activity. SHAPA allows researchers to develop an encoding vocabulary, to apply it to raw verbal protocol files, and to perform various types of data aggregation and data reduction. When performing verbal protocol analysis, researchers often try out different possible coding schemes before settling on the most effective one. SHAPA has been designed to support quick alterations to an encoding vocabulary and to support the rigorous statistical analysis of content and patterns (sequential data analysis) in protocol data. It is intended as an exploratory, as well as analytical, tool and has been designed to be easy for novices to learn and use, yet fluid and powerful for experts. A prototype version of SHAPA has already been used by a sample of researchers, and their experiences and requests have guided the programming of the present much more powerful version.

The analysis of sequential records is a research activity seen in many branches of the behavioral sciences, where it is given different names: sequential data analysis, process tracing, protocol analysis, interaction analysis, and various others. However, what all of these methods have in common is a concern with preserving sequential relations in data and with finding a succinct representation for the behavior thus recorded. A particular concern in our engineering psychology laboratory has been the analysis of verbal protocols, which are one type of sequential data. The focus of our research is the role of the human operator in complex sociotechnical environments, such as aircraft cockpits, emergency response centers, and industrial control rooms. In such environments, the human is more and more a supervisor and occasional troubleshooter rather than a direct controller. As a result, there is less overt behavior to observe as the human carries out his or her tasks. Thus, an increasingly important technique for research in this area is verbal protocol analysis. This paper will describe a PC-based interactive software program that we have developed, SHAPA,¹ which helps the researcher perform verbal protocol analysis.

There has been a highly active debate over the last 15 years about the validity of verbal reports (Bainbridge, 1979; Ericsson & Simon, 1984; Nisbett & Wilson, 1977; Praetorius & Duncan, 1988). We would argue that Ericsson and Simon (1984) have performed an important service by identifying for researchers the conditions under which verbal protocols might be considered to be valid or invalid. We assume that researchers using SHAPA have educated themselves in the substance of this debate and understand the extent to which the verbal data they have collected might be considered valid or invalid. Strictly speaking, an aid to protocol analysis such as SHAPA enhances validity only to the extent that it increases the reliability of encoding. Nonetheless, SHAPA has been designed to help researchers use verbal protocols *as data*, in the sense that Ericsson and Simon (1984) advocate. How this is achieved will become clearer as the software is described.

In the remainder of the paper, the need for SHAPA and the nature of the program will be covered. First, a short background of other verbal protocol analysis software environments will be given, and the background of SHAPA itself will be presented. Second, the design philosophy behind SHAPA will be discussed. Third, there will be a description of the functional purpose of SHAPA and an account of how the software supports heuristic activities during protocol analysis, as well as how the statistical information it provides might be used. Fourth, a fuller and more technical description of the features of the program will be given, organized around the menu structure of the

The early development of SHAPA was supported by an IBM EXCEL grant to P. M. Sanderson and the development of Version 2.01 was supported by a grant from NASA-Ames Research Center to P. M. Sanderson (Grant Monitors Everett A. Palmer III and Sandra Hart). Correspondence should be addressed to P. M. Sanderson, Department of Mechanical and Industrial Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801 (e-mail: psanders@psych.uiuc.edu).

software itself. Finally, some of the uses to which SHAPA has been put will be outlined.

PAST AND PRESENT VERBAL PROTOCOL ANALYSIS SOFTWARE

Protocol Analysis Software

As many researchers have noted, verbal protocol analysis is an immensely time-consuming activity. As a result, it is very rare for thorough, formal verbal protocol analyses to be attempted and reported, particularly analyses that summarize the data from many subjects. Over the last 20 years, there have been various attempts to automate verbal protocol analysis, or at least to make it interactive with a computer, in the hope of speeding and simplifying it. Many laboratories around the world have developed tools (or prototypes of tools) for performing verbal protocol encoding and offering some degree of analysis. At Carnegie-Mellon University, Waterman and Newell (1971, 1973) developed PAS I and PAS II, which were attempts at complete automatization, all the way from natural language parsing through to the generation of a "problem behavior graph," or graphical description of the subject's thought processes. PAS I was designed for cryptarithmic problems; PAS II was an attempt at developing a context-free automatic protocol analysis environment. Some years later, Bhaskar and Simon (1977) developed SAPA, a context-specific program for interactively coding protocols from subjects solving thermodynamics problems. Ericsson and Simon (1984) themselves developed MPAS, a context-free interactive aid to protocol analysis, whose notable feature was that it enabled researchers to code protocol segments in a random order, thus eliminating possible biases in encoding arising from inappropriate use of context. One of the most recent such tools used at Carnegie-Mellon is PAW, developed by Fisher (1988). PAW is a context-free interactive protocol analysis environment that includes a variety of statistical reports.

Interactive verbal protocol analysis programs have started to appear elsewhere as well. For example, VPA has been developed at IBM (Lueke, Pagerey, & Brown, 1987). PAP, a prototype of a verbal protocol analysis tool, has been developed at the Microelectronics and Computer Technology Corporation (Poltrick & Nasr, 1989). In addition, there are many other related tools that have been developed, or are in development, for video coding, content analysis, and discourse analysis (Laws & Barber, 1989; Mackay & Tatar, 1989; Sanderson, James, Watanabe, & Holden, 1990).

SHAPA

The development of SHAPA has been an ongoing project at the University of Illinois for some years (Sanderson, James, & Seidler, 1989). SHAPA is different from the software tools described above, because it explicitly makes the connection between verbal protocol analysis and powerful statistical tools typically used to analyze nonverbal sequential data, such as lag sequential analysis. In this sense, it has some features borrowed from

programs developed specifically for performing sequential data analysis, such as ELAG (Bakeman, 1983), MADA and ODAS (Kineapple, 1987), ANSEC and ASR (Quei & Estany, 1984), and the recent SATS (Yoder & Tapl 1990). The functional purpose of SHAPA for the researcher, and some of the philosophy behind its design will now be given.

This paper reports the current capabilities of SHAPA. A prototype, Version 1.2, was completed in 1989 (Sanderson et al., 1989) and was distributed to about 70 users worldwide. This paper describes SHAPA Version 2.0. This is a greatly extended version of the earlier prototype. It includes many features added in response to user requests and offers far more sophisticated file handling. It provides various statistical analyses for reliability and lag sequential analysis, in addition to frequency count. SHAPA Version 2.01 also represents a complete overhaul of the code of Version 1.2. In our opinion, it now represents a completed product. The design philosophy and functional purpose of SHAPA will now be examined.

DESIGN PHILOSOPHY

The principal design goal behind the development of SHAPA has been that the program should allow researchers the greatest degree of contact with their protocol data when developing an encoding vocabulary for it, encoding it, and analyzing it. In our experience, some of the currently available tools for performing sequential data analysis have highly modal interfaces, sometimes designed around batch commands, or they may be written for relatively uncommon machines. Of course, when these come from well-established laboratories where there are standard laboratory practices for collecting objective sequential data, this is understandable. Nonetheless, such features make it difficult for newcomers to the software to get started. More importantly, such interfaces discourage exploration and visualization of the data. In designing SHAPA for verbal protocol analysis, we have made an effort to avoid such problems. After all, verbal protocol analysis requires a researcher to study the verbal data very closely, and a flexible, fully interactive software environment that supports exploration becomes a necessity rather than something merely to be desired.

Software tools for performing verbal protocol analysis are less available than are the more general sequential data analysis packages referred to above. Some verbal protocol analysis software tools are highly specialized for a certain domain, some are prototypes only, some have come slowly to completion, and some are not well known because they have been used in-house only. In designing and developing SHAPA, we have made an effort to make the tool as general as possible and to reach out to potential users. First, we have wanted to make our colleagues aware of the possible advantages of performing protocol analysis with an interactive software tool. Second, we have wanted to collect users' opinions and experience; this was particularly valuable when developing SHAPA Version 2.01.

FUNCTIONAL PURPOSE OF SHAPA

Verbal Protocol Analysis Steps

How might researchers use SHAPA to analyze verbalizations, and what might they achieve through doing so? Researchers wishing to perform verbal protocol analysis typically start with audio or video tapes of verbalizations. Before using SHAPA, the recorded verbalizations need to be transcribed and segmented into lines using a word processor or editor, after which they can be read into SHAPA. At this point, SHAPA provides a software environment where researchers can carry out the following functions:

1. Make changes to the segmentation of the transcribed ("raw") protocol, if desired.
2. Examine the raw verbal protocol using different screen layouts.
3. Develop an encoding vocabulary (i.e., a set of labels, codes, or categories) for the protocol using a relational, or "predicate," notation.
4. Encode the protocol by associating selected lines of the protocol with a predicate chosen from the encoding vocabulary and replacing predicate arguments with further codes.
5. Perform various types of summary analyses on the protocol encoding.

Researchers are not restricted to this sequence, however. Verbal protocol analysis is a highly iterative process, and the researcher may wish to change the encoding vocabulary after performing summary analyses or to change the way the protocol is encoded. An important goal when designing SHAPA was to allow users to focus on their research problem when going through these iterations, rather than on the mechanics of using the interface. Thus, we have taken pains to make SHAPA easy to learn and operate for novice users, while affording the more experienced user a very fluid, natural interface. This supports the iterative nature of protocol analysis, where the researcher must engage in a considerable amount of activity but must never lose his or her intellectual momentum.

SHAPA has been designed to enhance researchers' contact with the meaning of their data both during encoding (when thinking heuristically) and during analysis (when thinking in more rigorous statistical terms). Just how this is achieved will now be described. (A fuller treatment of the features mentioned below will be given in the Program Operation section.)

Heuristic Support

Much of the heuristic power of SHAPA comes in the support for the process of developing an encoding vocabulary, for changing that vocabulary in midstream, and for viewing the data in different ways.

An important heuristic feature is the ability to make quick changes to the encoding vocabulary as encoding gets under way. If the structure of an encoding vocabulary item, code, or encoding category is altered, then this change is reflected wherever the vocabulary item has been used

in the encoding itself (see Predicates section below). The researcher has far less work to do in order to implement this change than would be the case when encoding by hand. As a result, the researcher's attention can remain focused on the *purposes* of the change to the vocabulary rather than on the mechanics of the change, and the researcher can more quickly judge whether the change has been effective.

There are different ways of viewing the raw protocol, and this encourages exploration. Different screen layouts are offered, so that the researcher can see encodings interlaced with raw protocol lines (as in Figure 1) or, alternatively, separated from them so that the raw protocol might be on the left of the screen and the encodings on the right (or vice versa) or the raw protocol at the top of the screen and the encodings on the bottom (or vice versa) (see Layout section below). Different layouts are convenient at different points in the protocol analysis process and can highlight different aspects of the data. Furthermore, users can use the Tab key to "indent" their protocol encodings in order to better visualize the structure of verbalization (as, for instance, is often used in GOMS models; see Card, Moran, & Newell, 1983).

SHAPA's data summary capabilities need not be confined to the end of an encoding session when final results are gathered; they can be used at any point during the encoding process. Researchers can ask SHAPA to collect and display all parts of the raw protocol that have been encoded with a specified predicate. This allows researchers to judge whether they are applying their encoding vocabulary reliably (see Predicate Instances within the section on Reports below). In addition, researchers can ask SHAPA to generate an overall summary of how the encoding vocabulary has been used up to the present point (see Value Lists). This helps to eliminate redundancies in the use of the encoding vocabulary or conflicting usages of terms; again, it is an important aid to developing a well-structured encoding vocabulary that is reliably used. Overall, the ability to use data reports at any point during encoding can reveal emerging patterns and can encourage the researcher to think of better ways to encode so that the regularities in the data are highlighted. However, deciding how much context and foreknowledge of results should be used during encoding is an important scientific judgment (Ericsson & Simon, 1984), and the researcher should establish standards for each program of investigation.

A feature of SHAPA that promotes exploration of the protocol data is the ability to "filter" out certain parts of the encoding before performing data summaries or reports. For example, data summaries might be confined to certain sections of the protocol, such as the beginning or end. Alternatively, data summaries might be confined to a subset of the encoding vocabulary only, if, for example, the researcher has doubts about the appropriateness or relevance of some of the predicates in the encoding vocabulary.

As well as providing different ways of examining vocabulary usage, the above features allow researchers to

File	Encode	Predicates	Search	Layout	Report
Line 23	Col 5	MBOENF7<-MBOENF7.EN1	Insert	*	108 lines --
21	keep taking that	ENCOURAGER(kg)			
22	now it's going to be going straight down	STATUS(feature,old)			
23	and then go straight down	COMMAND(move)			
24	and then it, like, stops	STATUS(feature,old)			
25	and there is a fork going off to the left	STATUS(feature,new)			
26	and a little bit up and to the left	DESCRIPTION(orientation)			
27	by a grange.	DESCRIPTION(orientation)			
28	You'll want to take that going to the left.	COMMAND(tr)			
29	You pass a hill	STATUS(self,)			
30	and you keep following that main road	ENCOURAGER(kg)			
31	you go off to the left a little bit	STATUS(self,)			
Syntax: COMMAND(<WHAT>) F5 Join F7 Split Alt-A Toggle Active					

Figure 1. SHAPA's screen, showing the pull-down menu options and a protocol in the process of being encoded. The help line is at the bottom.

move from thinking in global terms about their protocol to thinking in detailed terms about encoding vocabulary usage and vice versa, as the situation requires. In addition, the ability to conduct searches on the basis of key words or phrases with a special theoretical significance can help researchers narrow their focus when needed, such as when pursuing a specific hypothesis about the verbal data (see Search section).

Finally, researchers may sometimes wish to encode the same raw protocol in several different ways. Multiple encoding files can be created for a given raw protocol and can be stored separately from each other (as independent files on disk). These different encoding files might represent different theoretical approaches to the data—for example, a comparison of the descriptive or explanatory power of two theories of learning in a concept discovery task. Alternatively, they might represent the pursuit of qualitatively different hypotheses about the data (examining the propositional structure of utterances *versus* the level of speech act mitigation) or different levels of abstraction (at a low level of abstraction, a researcher may be interested in the physical objects or events mentioned, whereas, at a higher level of abstraction, a researcher may be more interested in how the subject goes about reaching his or her goals). Finally, the different encoding files might simply represent a researcher's different attempts at encoding a raw protocol file, where fresh attempts were saved under new names. In principle, the researcher can maintain as many different encoding files for the same raw protocol file as desired.

Statistical Support

Researchers will often collect verbal protocols in the context of an experimental design and will have hypotheses on how one group of subjects might perform compared with another or how the same group of subjects might perform in different situations. SHAPA provides summary data for individual subjects through its data aggregation and data analysis routines. In various different ways, these routines summarize the content of the protocol encoding and highlight any patterns that might emerge in it. Such data can be collected across a set of protocols and can then be submitted to conventional statistical analyses. When using SHAPA, researchers should be able to give more protocols a full encoding and analysis than would otherwise be possible. This is because, in principle, SHAPA allows researchers to converge upon a successful coding scheme more efficiently, to encode protocols more reliably, and to go through the mechanics of encoding many protocols more quickly.

For example, the researcher might be interested in examining the use of linguistic "clarifications" by a speaker talking to listeners who give a relatively high amount of verbal feedback compared with listeners who give relatively little verbal feedback. The researcher might hypothesize that the verbal protocols of speakers communicating with high-feedback listeners would show a lower proportion of linguistic clarifications than the protocols of speakers communicating with low-feedback listeners. The content analysis reports in SHAPA provide statistics that can become data points in further statistical

tests. It is up to the researcher to decide whether the data are better expressed as proportions and, if so, what the appropriate denominators should be for such calculations. In the example above, SHAPA would provide the necessary statistical output for calculating the proportions from each speaker's protocol. These results would be tabulated and tested with a *t* test, an analysis of variance (ANOVA), or any other suitable test, in a separate statistics package.

SHAPA's pattern analytic reports (see under Reports below) also provide statistics that can be collected for a sample of subjects and then analyzed in conventional statistical packages in much the same way. The researcher can take the results of any of the statistical reports and, after making any necessary adjustments to ensure a proper comparison, can use them in a wide variety of tests. Such a capability takes Ericsson and Simon's (1984) exhortation to use verbal protocol data as *data* to its logical conclusion. With SHAPA, the researcher can more quickly and reliably use verbal protocols to test hypotheses using the traditional scientific method, if this is desired. However, this is not necessarily the end point for every scientific investigation of verbal protocols that might be conducted with SHAPA.

PROGRAM OPERATION

In this section, more formal details of the operation of SHAPA are given. Technical information is provided first, followed by a discussion of SHAPA's operations.

Technical Information

SHAPA Version 2.01 is written in Turbo C and runs on IBM PCs with a minimum of 512K of RAM. It is also designed to run on compatibles and clones: we have successfully run it on HP Vectras, Compaqs, Olivettis, and Zeniths. Protocol files are not loaded into memory in their entirety, so their size does not directly affect memory limits. The only limit is that they must not be larger than 64K (after this, multiple-file routines can handle longer protocols). Raw protocol files must consist of segmented verbalizations where each segment, or line, ideally consists of no more than 75 characters. SHAPA works best with EGA displays but will work quite adequately with CGA displays. Statistical analyses on average-sized files can generally be performed within only a few seconds on the older generation of IBM PCs, whereas statistical analyses can be completed in under a second on faster processors. Of course, this depends on the length of the encoding being analyzed. The software is compiled to run with or without a math coprocessor.

We have produced a manual of about 50 pages. The first section of the manual is a step-through tutorial, and the second section is a general reference to SHAPA's features. SHAPA comes with a collection of demonstration files, including the two used in the present paper.

Menu Structure

In the following sections, we will provide more detail about the operation of the program and its software fea-

tures. We will illustrate these features using a verbal protocol taken from a study of language used in navigational command under different conditions of feedback from those receiving commands (Moray & DeSalvo, 1991; Sanderson, 1990).

SHAPA is driven by pull-down menus on a menu bar at the top of the screen. Figure 1 illustrates how the SHAPA interface might appear to a researcher halfway through an encoding session. The menu labels are "File," "Encoding," "Predicates," "Search," "Layout," and "Reports." All options on these menus are accessible through cursor keys (there is no provision for the use of a mouse) or through power-key equivalents. The latter feature allows experienced users to avoid menu operations.

A status line at the bottom of the SHAPA screen gives information about the current mode of the program and options currently available. It also lists the commands for the most important options given the current mode. For example, it gives information about the syntax of the predicate the program cursor is currently highlighting or the commands available when changing the syntax of predicates.

In the sections that follow, SHAPA's operation and features will be described using the pull-down menu structure as a guide. Description will proceed from left to right across the menu bar (see Figure 1), with the exception that Predicates and Encoding will be interchanged (Encoding is easier to understand if Predicates have been described beforehand).

File

SHAPA assumes that once encoding has started, each raw protocol will have one or several encoding files associated with it. Each of these encoding files is logically linked to the raw protocol file on which it was based. SHAPA allows researchers to keep many raw protocol files and their associated encoding files "on hand," or "current," during an encoding session. However, only two files at a time can be displayed, and these two files must be associated with the same raw protocol or be a raw protocol and one of its associated encoding files. Changing which files are displayed is a very simple operation involving cursor movements among file names in a menu of current files.

The file menu also allows several program parameters to be set and stored, such as margin, page-length, and character-size instructions for a printer. Additionally, international users of the SHAPA prototype often need extended character sets (for accents or non-English characters) and may also need right-to-left orientation of the raw protocol display. For example, users in Israel require an extended character set to accommodate Hebrew characters, as well as a right-to-left display. These parameters can be set within the program so that Hebrew raw protocol can be immediately loaded into SHAPA (however, encoding and all other text remain in roman left-to-right characters).

File		Encode	Predicates	Search	Layout	Report		
Line 1	Col 1		MBOENF7<-MBOENF7.EN1		Insert	*	7 lines	--
Short	Syntax							
C	COMMAND	(<WHAT>)						
O	OPTIONS	(<TYPE>)						
D	DESCRIPTION	(<FOCUS>)						
B	BREAK-OFF	(<FUNCTION>)						
S	STATUS	(<FEATURE> , <OLD/NEW>)						
E	ENCOURAGER	(<TYPE>)						
		()						
<hr/>								
Edit Predicates			Alt-E Encode	Tab Filter	^Y Delete Predicate	Alt-F Find		

Figure 2. The predicate screen, showing a sample vocabulary that has been entered by the researcher. The help line now has contents different from those in Figure 1.

In many cases, researchers wish to use the same encoding vocabulary, or set of predicates, for encoding many raw protocol files. Researchers can borrow an encoding vocabulary from a previously encoded file rather than having to type it in again.

Predicates

The encoding vocabulary is entered and edited on a separate screen, which is accessed through the Predicates option on the menu bar. An encoding vocabulary consists of predicates, each of which is composed from a main label (the predicate) and a list of arguments that can be replaced by constants (see Figure 2, for an example). For example, the user might enter the predicate STATUS (**<FEATURE>**,**<OLD/NEW>**) and use it to encode all statements in the raw protocol where the subject is talking about a current state of affairs. The **<FEATURE>** and **<OLD/NEW>** arguments would be replaced by information relating to the type of feature and whether it had been mentioned before. "Landmark" could replace **<FEATURE>** and "old" could replace **<OLD/NEW>** to give STATUS(landmark,old). How the content is translated into constants is arbitrary, and the subject's verbalization might as readily have been encoded as STATUS(l,o) or as any of a number of variants.

The Predicate screen allows users to create as many predicates as they need: the screen will scroll to fit them all. The user should determine a short form for each predicate, such as "C" for COMMAND (see Figure 2). This encoding vocabulary can be altered at any point. New predicates can be added and old predicates deleted. Ad-

ditions and deletions can be made to the argument list of each predicate, and the order of a predicate's arguments can be changed. All changes are echoed in the encoding that the researcher has performed so far.

The predicates and their arguments must be carefully chosen to reflect the researcher's theory. They must also allow the information in the raw protocol to be represented in a way that makes it easy to obtain answers, using the various Report options, to the research questions motivating the research. The reader is referred to Ericsson and Simon (1984) and to Sanderson et al. (1989) for further discussion of how an encoding vocabulary is developed, altered, used, and maintained.

As mentioned above, predicates can be borrowed from another encoding file, or predicates from another encoding file can be added into an existing set of predicates. When many protocols are being analyzed from one experiment, this saves the user the labor of retyping the encoding vocabulary for every new encoding file created.

Encoding

Choosing the Encoding option on the menu bar simply returns control to the encoding screen. This is where the researcher uses the predicate encoding vocabulary to encode the information in the raw protocol. The encoding screen displays either a raw protocol and an encoding file (see Figure 1) or, if desired, two encoding files that are based on the same raw protocol. The latter is necessary before carrying out certain analyses that are based on two encodings, such as interencoding reliability measures, but it may also be needed for heuristic purposes. There are

various ways that the data in the two files can be displayed with respect to each other, and these will be described further under the Layout heading.

The cursor can be active on only one file at a time, but the user can very quickly switch the cursor to the other file. The types of editing operations possible are different depending upon whether the cursor is positioned on the raw protocol file or an encoding file. The only editing operations possible on raw protocol files are the splitting of lines into two or the joining of two lines into one. In contrast, a variety of features are available when the cursor is on an encoding file.

The process of encoding is facilitated by a "structure editor," which relieves the researcher of much of the burden of managing the syntax and punctuation of the predicates. First, if the user types in the short form for a predicate, as soon as the Enter key is pressed the line "explodes" to show the full syntax of the predicate. The full syntax of the predicate is also displayed in the help line at the bottom of the screen. The researcher then has a predicate "shell" into which the specific details of the present case can be entered. Second, the management of predicate punctuation is handled by SHAPA. This means that the researcher can concentrate on the choice and entry of constants for the arguments, and the correct punctuation of the predicate will automatically be maintained.

In SHAPA, it is not necessary to encode every line of the raw protocol. Researchers can encode at any level of abstraction they wish. Thus, if verbalization about a particular topic runs for several segments, or text lines, only the first line need be encoded and the remainder can be left unencoded. SHAPA provides different ways of handling unencoded lines, which will be described under the Reports heading.

A further feature of the encoding screen is the ability to indent encodings using the Tab key. This allows researchers to structure the encoding to highlight the flow of control or of the passage from one episode to the next (see Figures 1 and 3). These indentation settings are stored when the file is saved.

Search

SHAPA allows users to search for occurrences of a text string, predicate, or constant within a predicate. This is useful when examining a large protocol with a specific question in mind, such as how a navigational guide may convey spatial orientation information. For example, by searching the raw protocol for occasions where the guide used such phrases as "right," "left," "ahead," "far," "near," "further on," and so on, the researcher might focus quickly on at least some of the episodes in which such orientation information was being used. The ability to search is also useful when examining how a certain predicate has been used and when standardizing encoding procedures. For example, if all the STATUS and all the DESCRIPTION predicate encodings are listed with their associated raw protocol, are the two sets conceptually distinct and consistently encoded? The Search menu

also enables the user to quickly move the cursor to a given segment with a "Go To" (line number) option.

In addition, SHAPA allows researchers to encode a raw protocol in a random segment order. The user can make the cursor jump to a randomly chosen segment by typing a certain function key. This makes it less possible to use context when encoding: the use of context is, in some cases, considered inappropriate (see Ericsson & Simon, 1984).

Layout

There are different ways that two files in SHAPA's memory can be displayed with respect to each other. Researchers have different preferences about layout, depending upon the stage of encoding they have reached, the type of encoding vocabulary they are using, the nature of the verbal material they are dealing with, and so on. If there are two files in memory (File A might be a raw protocol and File B an encoding file), then they may be displayed on alternate lines down the screen with the raw protocol first (ABABABAB...), as in Figure 1, or the encoding file first (BABABA...). A second option is that File A may be displayed in the upper half of the screen and File B in the lower half, and vice versa. A third option is that File A may be displayed on the left of the screen and File B on the right, or vice versa. Finally, File A or File B alone might be displayed. If the length of an encoding or a raw protocol segment is too long for its field, given the current display format, then the line can be made to scroll so that the remainder can be seen. The same options as just described also hold when there are two encoding files being displayed.

Reports

SHAPA offers several different types of data reduction techniques, which we call *reports*. These will be outlined below. Many reports have options that must be set before the report can be run. In addition, there are some options that can be set for a given encoding file that will hold for any report subsequently performed in that session: this we call *filtering*.

Filtering. The concept of filtering is very important and is one of the major features of Version 2.01. The researcher can limit the line numbers included in the analysis (e.g., in a 200-line file, the researcher may be interested in seeing results for lines 55 to 155 only). The researcher can also limit the levels of indentation analyzed (e.g., the encodings on the left margin [0] and first indent [1] only of the 10 possible). If encodings placed on different indents have represented different levels of abstraction, different levels of control, or even different people speaking, then clearly this allows the researcher to narrow the scope of the analysis in ways that are potentially very informative.

Furthermore, filtering provides three different ways of treating lines that have not been encoded. Such lines are "null codings," and they appear as encoding file lines consisting of "()" rather than a predicate and arguments.

First, such null codings can be skipped—that is, treated as nonexistent events. In this case, the sequence “closes up” over them, so that the sequence A B () C would become A B C, with C as the third event. Second, null codings can be included as important events in their own right. In the sequence A B () C, () would be the third event and would be reported as a () event. Third, null codings can be treated as breaks in a sequence—that is, no transitions are recorded to, from, or across null codings. In the sequence A B () C, the sequence would stop at B and resume at C without memory of A, B, or ().

A final filtering feature in SHAPA is the ability to exclude some predicates from an analysis, if wished. This can be done by marking the unwanted predicates on the predicate screen. This effectively sets these predicates to () so that they can be handled in any of the ways just described. All these filtering options are set with respect to a single encoding file and remain in place until they are reset or until the program is terminated.

In some cases, it may be that the data from a single observational period have been broken into several contiguous files (running one off the other). When the “multiple files” option on the Report menu is activated, such multiple files can be handled as if they were one file.

The filter options can be left at their default values or set for special needs, as described above. Then the data from the encoding file can be analyzed using any of the different types of report offered (*content analytic* or *pattern analytic*). Some of these reports involve options that must be set before they can be run. The results of the reports are sent to the screen and can then also be directed to the printer or to an output file on disk.

Content analytic reports. The content analytic reports are (1) Encoding, (2) Value Lists, (3) Predicate Instances, and (4) Cross Reliability.

Encoding. This report lists an encoding and/or its raw protocol file in various ways. Researchers can choose whether or not they want the segment line numbers reported, the raw protocol reported, the indentations preserved, and the predicate vocabulary listed at the top of the listing. Figure 3 gives an example of a printout with segment numbers but no protocol reported, indentations preserved, and predicate vocabulary included. It shows that the communication of navigational information tends to be organized around COMMAND (“go up the hill”) and STATUS (“you are now at the top of the hill”) verbalizations.

Value Lists. This report is a classic content analysis that prints out statistics about encoding vocabulary usage. This includes a count of how often predicates, constants within predicate arguments, and (if wished) predicates within other predicates (embedded predicates) were used in an encoding. The report provides the option of reporting any embedded predicates in the context of the parent predicate in which they were embedded or at the end of the report in a separate breakdown. Figure 4 gives an example (no embedded predicates were used in this encoding).

```
MBOENF7.EN1 (MBOENF7)
  Null codings skipped
  98 encoding lines reported out of 108 lines in file
```

```
COMMAND(<WHAT>)
OPTIONS(<TYPE>)
DESCRIPTION(<FOCUS>)
BREAK-OFF(<FUNCTION>)
STATUS(<FEATURE>,<OLD/NEW>)
ENCOURAGER(<TYPE>)

1 COMMAND(tr)
2   DESCRIPTION(orientation)
4   STATUS(feature,old)
5   STATUS(self, )
6   COMMAND(mwrtf)
7   COMMAND(mwrtf)
8   STATUS(self, )
9 COMMAND(move)
10  BREAK-OFF(abrupt)
11  STATUS(self, )
12    BREAK-OFF(correction)
13  STATUS(self, )
14    BREAK-OFF(back-up)
15  STATUS(feature,old)
16  STATUS(feature,old)
17  STATUS(self, )
18  STATUS(feature,old)
19  STATUS(feature,new)
20  STATUS(feature,new)
21  ENCOURAGER(kg)
22 STATUS(feature,old)
23  COMMAND(move)
24  STATUS(feature,old)
25  STATUS(feature,new)
26  DESCRIPTION(orientation)
27  DESCRIPTION(orientation)
28 COMMAND(tr)
29  STATUS(self, )
30  ENCOURAGER(kg)
31  STATUS(self, )
32  STATUS(feature,old)
34  STATUS(self, )
35  ENCOURAGER(kg)
36 STATUS(self, )
37  DESCRIPTION(object)
38  DESCRIPTION(object)
39  COMMAND(move)
40  STATUS(feature,old)
41  ENCOURAGER(kg)
42 STATUS(feature,new)
43  OPTIONS(roads)
45  STATUS(feature,old)
46  STATUS(feature,old)
48 COMMAND(tr)
49  STATUS(self, )
```

Figure 3. An Encoding printout. The encoding vocabulary is listed at the top. To save space, this printout shows only part of the encoding.


```

MBOENF7.EN1 (MBOENF7)
  Null codings skipped
  98 encoding lines reported out of 108 lines in file

Breakdown of Predicate Usage

BREAK-OFF(<FUNCTION>) [7]
  <FUNCTION>: abrupt [4], back-up [2], correction

COMMAND(<WHAT>) [18]
  <WHAT>: move [3], mwrtrf [7], tr [8]

DESCRIPTION(<FOCUS>) [13]
  <FOCUS>: object [5], orientation [8]

ENCOURAGER(<TYPE>) [10]
  <TYPE>: kg [10]

OPTIONS(<TYPE>) [5]
  <TYPE>: roads [5]

STATUS(<FEATURE>, <OLD/NEW>) [45]
  <FEATURE>: feature [24], self [21]
  <OLD/NEW>: [21], new [11], old [13]
    
```

Figure 4. A Value List report printout. This shows how many times each predicate, and each constant within a predicate, has been used. (Within COMMAND, "mwrtrf" stands for "move with respect to feature" and "tr" stands for "take road." Within ENCOURAGER, "kg" stands for "keep going.")

Predicate Instances. Researchers may wish to focus on the usage of a particular predicate (such as DESCRIPTION) or of a particular constant within a predicate [such as "object" as in DESCRIPTION(object)]. This report prints out the raw protocol and the encoding for all such segments, so they can be examined in isolation.

Cross Reliability. This report performs a reliability check across two encoding files based on the same raw protocol. A table reporting the match frequencies is printed, and Cohen's kappa (Cohen, 1960) and associated statistics are also reported. Figure 5 gives an example. The Cross Reliability report also allows the researcher to determine how similar two different encoding vocabularies are to each other (in this case, Cohen's kappa is not relevant). For example, one researcher's GUESS predicate might be another's HYPOTHEZIZE.

Pattern analytic reports. SHAPA's pattern analytic techniques include (1) Transition Matrices, (2) Lag Sequential Analysis, and (3) Frequency of Cycles.

Transition Matrices. First-order, second-order, and third-order transition frequencies between predicates are reported, with their marginals. This allows the researcher to determine whether or not there are strict sequences of short lengths in the encoding.

Lag Sequential Analysis. Lag Sequential Analysis (LSA) reports the frequency with which a certain predicate occurs at various removes just before and just after all instances of another predicate. This allows researchers to detect sequential dependencies in the data, but those de-

pendencies are not as rigidly tied to a sequence as with transition matrices.

The LSA report in SHAPA requires one predicate, or a constant within a certain predicate, to be identified as the "key" for the analysis. All the remaining predicates (those at least that have not been filtered out) become "targets." The data can be presented in histogram form and/or statistical form. The descriptive statistics provided are frequency counts of the presence versus absence of the key against the presence versus absence of each of the targets for each lag being considered. Statistics provided to test the strength of dependencies are Allison and Liker's (1982) "corrected" z score, Faraone and Dorfman's (1987) "Markov" z_m score, Pearson's r , and Yule's Q (Reynolds, 1984). Allison and Liker's (1982) z score is corrected from the original Sackett (1979) formula, and Faraone and Dorfman's (1987) z_m score corrects for autocorrelation at Lag 1. As Yoder and Tapp (1990) have pointed out when discussing their SATS program, Pearson's r and Yule's Q are important because they are measures of dependency that are independent of the n on which they were computed. This allows them to be used in t tests and ANOVAs for cross-subject and cross-condition comparisons, if desired. This feature is important if verbal protocol data are indeed to be used as data in the sense that Ericsson and Simon (1984) advanced. The two mea-

```

Vertical: MBOENF7.EN1 (MBOENF7)
  Null codings skipped
  98 encoding lines reported out of 108 lines in file

Horizontal: MBOENF7.EN2 (MBOENF7)
  Null codings skipped
  101 encoding lines reported out of 108 lines in file
    
```

Relative Frequencies

	---MBOENF7.EN2-----							
MBOENF7.EN1	C	O	D	B	S	E	MAT	TOT
COMMAND	13	.	.	.	3	2	18	18
OPTIONS	.	3	.	2	.	.	5	5
DESCRIPTION	1	1	8	.	3	.	13	13
BREAK-OFF	.	.	.	4	1	.	5	7
STATUS	3	.	1	2	37	.	43	45
ENCOURAGER	.	.	.	.	.	10	10	10
MATCHED	17	4	9	8	44	12	94	
TOTAL	17	5	13	8	46	12		

```

Observed probability = 0.7979
Chance probability   = 0.2824
Cohen's kappa       = 0.7183
Maximum kappa       = 0.9111
Standard error k    = 0.0577
Standard error Z    = 0.0647
Z-score of kappa    = 11.10
    
```

Figure 5. A Cross Reliability analysis printout. The fact that most observations fall on the left diagonal suggests that there is reasonable, but not perfect, reliability between the two encoding files. The fact that the predicates matched may be less than the total number of predicates of that type in each file means that the remainder were matched by no predicate at all, or "0".

sures differ in that Pearson's r requires strict perfect association for the statistic to reach +1, whereas Yule's Q requires only weak perfect association to reach +1 (Reynolds, 1984).

SHAPA users can specify (1) if they want to see LSA histograms, LSA statistics, or both, (2) the maximum lag to which the LSA should be taken, and (3) whether key/target instances that have another instance of a key or target predicate between them should be counted or not. For example, in the sequence ABAACBBDAB, when A is the key and B the target, there are various ways one might search for cases in which A is followed at some remove by a B. There are three cases where the transition is broken by neither another A nor another B, 11 cases where the A-to-B transition contains other examples of A and/or B, four cases where another A but not another B is contained within the transition, and four cases where another B but not another A is contained within the transition. Of course, in the case just given, the lags are of different lengths, whereas, in LSA, the user would be concerned with the count at a fixed lag. In addition, the count for a sequence may differ considerably, depending upon how null [()] codes are handled.

Figure 6 shows part of a graphical and statistical LSA printout. It shows that this navigational guide tends to provide detailed descriptions of geographic features immediately before issuing a command to move, presumably in order to reduce uncertainty about the expected current location before continuing. The trend seen in the histogram is statistically significant.

Frequency of Cycles. The final pattern analytic report in SHAPA is Frequency of Cycles. This is a technique reported by Fisher (1988), which allows researchers to see loose patterns in their data. The researcher specifies a key predicate, such as COMMAND. The program finds each instance of COMMAND in the encoding and prints out the names of the predicates that lie between that instance of COMMAND and the next one. If behavior tends to cycle, then the patterns of predicates should be quite similar, and there may be multiple instances of the most typical patterns. The value of this technique is that even if all patterns are different, if there are only small differences between them, it is quite likely that the human observer will detect the general pattern. The Frequency of Cycles report also lists the order in which the cycles occurred in the encoding, so that the researcher has an overview of the patterns in the whole protocol. This reveals whether the patterns themselves occur in cycles: a type of support for pattern recognition at an even higher level of abstraction.

Figure 7 shows an example. For this navigational guide, COMMANDs are usually followed by STATUS (which is a very frequent predicate in the first place and more likely than any other to do so). It can also be discerned that ENCOURAGER ("keep going") predicates tend to precede STATUS predicates but that DESCRIPTION predicates do not. This could be confirmed in separate Frequency of Cycle reports. In this way, the patterns in the data come to light.

```
MBOENF7.EN1 (MBOENF7)
Null codings skipped
98 encoding lines reported out of 108 lines in file
```

```
Search Key: COMMAND
Key predicate intervention Disallow
Target predicate intervention Disallow
```

Frequency of DESCRIPTION with respect to COMMAND

DESCRIPTION with respect to COMMAND

Lag	K=0 K=0 K=1 K=1				Pearson's r	Yule's Q	Corrected Z	Markov Z
	T=0	T=1	T=0	T=1				
-4	39	3	11	0	-0.1254	-1.0000	-0.9126	-0.9206
-3	50	3	10	2	0.1603	0.5385	1.2920	1.2816
-2	60	5	13	2	0.0779	0.2973	0.6969	0.6972
-1	73	7	11	6	0.2962	0.7010	2.9175	2.9404
1	69	11	15	2	-0.0222	-0.0891	-0.2182	-0.2199
2	56	9	13	2	-0.0058	-0.0218	-0.0520	-0.0520
3	45	8	11	1	-0.0759	-0.3233	-0.6123	-0.6074
4	35	7	10	1	-0.0858	-0.3333	-0.6248	-0.6303

Figure 6. Part of a Lag Sequential Analysis printout, showing the histogram and LSA statistics for one key-target combination. This analysis suggests that, for the present protocol, DESCRIPTIONs tend to fall immediately before COMMANDs. Normally information for all targets is provided.

Further uses of report data. Any report generated by SHAPA can be saved to disk as an ASCII file. It can then be used directly for other purposes or submitted to post-processing so that it can be read by other applications or statistics packages.

CURRENT USAGE OF SHAPA

The development of SHAPA would be a sterile exercise if researchers did not use it. Thus, its development has gone hand in hand with an effort to distribute copies of the prototype to interested colleagues. By October 1990, copies of SHAPA Version 1.2 had been distributed to about 70 colleagues in Canada, France, Italy, Germany, Australia, New Zealand, Israel, the U.S.A., and the United Kingdom. SHAPA is being put to a wide variety of uses in the fields of cognitive engineering, educational

MBOENF7.EN1 (MBOENF7)

Null codings skipped

98 encoding lines reported out of 108 lines in file

Cycle around COMMAND

17 cycles occur less than 25 length

Freq Cycle

```

A.  2  COMMAND -> COMMAND
B.  1  COMMAND -> DESCRIPTION -> STATUS -> STATUS -> COMMAND
C.  1  COMMAND -> DESCRIPTION -> DESCRIPTION -> ENCOURAGER -> STATUS -> BREAK-OFF -> OPTIONS -> DESCRIPTION -> COMMAND
D.  3  COMMAND -> STATUS -> COMMAND
E.  1  COMMAND -> STATUS -> STATUS -> DESCRIPTION -> DESCRIPTION -> COMMAND
F.  1  COMMAND -> STATUS -> STATUS -> STATUS -> BREAK-OFF -> COMMAND
G.  1  COMMAND -> STATUS -> STATUS -> ENCOURAGER -> OPTIONS -> ENCOURAGER -> STATUS -> STATUS -> BREAK-OFF -> STATUS
    -> STATUS -> STATUS -> STATUS -> DESCRIPTION -> COMMAND
H.  1  COMMAND -> STATUS -> ENCOURAGER -> STATUS -> COMMAND
I.  1  COMMAND -> STATUS -> ENCOURAGER -> STATUS -> STATUS -> STATUS -> ENCOURAGER -> STATUS -> DESCRIPTION
    -> DESCRIPTION -> COMMAND
J.  1  COMMAND -> STATUS -> ENCOURAGER -> STATUS -> OPTIONS -> STATUS -> STATUS -> COMMAND
K.  1  COMMAND -> STATUS -> ENCOURAGER -> STATUS -> ENCOURAGER -> DESCRIPTION -> OPTIONS -> DESCRIPTION -> COMMAND
L.  1  COMMAND -> STATUS -> DESCRIPTION -> STATUS -> STATUS -> STATUS -> DESCRIPTION -> COMMAND
M.  1  COMMAND -> BREAK-OFF -> STATUS -> BREAK-OFF -> STATUS -> BREAK-OFF -> STATUS -> STATUS -> STATUS -> STATUS
    -> STATUS -> STATUS -> ENCOURAGER -> STATUS -> COMMAND
N.  1  COMMAND -> BREAK-OFF -> OPTIONS -> COMMAND
    
```

Cycle Sequence:

B -> A -> D -> M -> E -> I -> J -> F -> H -> A -> M -> L -> K -> C -> D -> G -> D

Figure 7. Frequency of Cycles printout. The sequences A to N are the different cycles observed. The Cycle Sequence shows the order in which the sequences A to N occurred within the encoding file.

psychology, decision making, engineering and industrial design, architecture, and medical problem solving.

Of course, the ultimate test of our ideas about how verbal protocol analysis might be supported by an interactive tool is whether researchers interested in protocol analysis adopt it as a standard laboratory tool alongside their statistical and word-processing application programs. Many of our colleagues clearly found Version 1.2 useful. For example, SHAPA (with its Hebrew adjustments!) has been heavily used at the Technion in Israel to compare the verbal communication and problem-solving styles of intensive-care unit doctors and nurses (Gopher, personal communication, 1990). It has also been used at EG&G Idaho in a large study of flight skills in the cockpit (Hahn & Blackman, 1990). The Aeronautical Research Laboratory within DSTO in Melbourne, Australia, has used SHAPA for studying crew communication during surveillance missions (Manton & Hughes, 1990). Another aviation application is under way at NASA-Ames, where SHAPA is being used to look at how flight crews handle navigational tasks and navigational information (Hart, personal communication, 1990). SHAPA has also been used at the University of Southern California to examine the problem-solving processes of the human when required to program industrial robots (Rahimi & Azevedo, 1990). Many of these research programs are continuing with SHAPA Version 2.01. Within our own laboratory at Uni-

versity of Illinois (Engineering Psychology Research Laboratory), SHAPA has been used to examine the verbal protocols of teams of people carrying out navigational tasks with different constraints on fluency of communication between team members, for examining the domain knowledge of human controllers of thermodynamic processes, and for identifying different types of strategic behavior in a variety of laboratory problem-solving tasks (Sanderson, 1990).

The functionality and features of SHAPA Version 2.01 have been developed in reaction to the comments and suggestions of these and other users. In a very real sense, SHAPA represents "research on research"—research on how a certain type of data analysis might be, and actually is, carried out. We hope we have created a powerful and flexible tool, accommodating to researchers' needs and capable of handling the idiosyncracies of a wide variety of verbal data sets. As always, we will be intrigued to witness the further uses to which SHAPA will be put in the future.

Availability

We can make copies of SHAPA Version 2.01—plus the 50-page manual and supporting materials—available to interested colleagues. We ask for a check for \$10, which covers materials, postage, and handling. The check should be made out to the University of Illinois, and should be

sent to P. M. Sanderson, Department of Mechanical and Industrial Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801.

REFERENCES

- ALLISON, P. D., & LIKER, J. K. (1982). Analyzing sequential categorical data on dyadic interaction: A comment on Gottman. *Psychological Bulletin*, **2**, 392-403.
- BAINBRIDGE, L. (1979). Verbal reports as evidence of the process operator's knowledge. *International Journal of Man-Machine Studies*, **11**, 411-436.
- BAKEMAN, R. (1983). Computing lag sequential statistics: The ELAG program. *Behavior Research Methods & Instrumentation*, **15**, 530-535.
- BHASKAR, R., & SIMON, H. A. (1977). Problem solving in semantically rich domains: An example from engineering thermodynamics. *Cognitive Science*, **1**, 193-215.
- CARD, S., MORAN, T., & NEWELL, A. (1983). *The psychology of human-computer interaction*. Hillsdale, NJ: Erlbaum.
- COHEN, J. (1960). A coefficient of agreement for nominal scales. *Educational & Psychological Measurement*, **20**, 37-46.
- ERICSSON, K. A., & SIMON, H. A. (1984). *Protocol analysis: Verbal reports as data*. Cambridge, MA: MIT Press.
- FARAONE, S. V., & DORFMAN, D. D. (1987). Lag sequential analysis: Robust statistical methods. *Psychological Bulletin*, **101**, 312-323.
- FISHER, C. (1988). Advancing the study of programming with computer-aided protocol analysis. In G. Olson, E. Soloway, & S. Sheppard (Eds.), *Empirical studies of programmers 1987 Workshop* (pp. 198-216). Norwood, NJ: Ablex.
- HAHN, H., & BLACKMAN, H. (1990). *Verbal reports: A viable method for performance measurement in complex environments?* Manuscript submitted for publication.
- KINEAPPLE, K. (1987). The microanalytic data analysis package. *Behavior Research Methods, Instruments, & Computers*, **19**, 335-337.
- LAWS, J. V., & BARBER, P. J. (1989). Video analysis in cognitive ergonomics: A methodological perspective. *Ergonomics*, **11**, 1303-1318.
- LUEKE, E., PAGEREY, P. D., & BROWN, C. R. (1987). User requirements gathering through verbal protocol analysis. In G. Salvendy (Ed.), *Cognitive engineering in the design of human-computer interaction and expert systems* (pp. 523-528). Amsterdam: Elsevier Science.
- MACKAY, W. E., & TATAR, D. G. (1989). Introduction to the special issue on video as a research and design tool. *SIGCHI Bulletin*, **21**, 48-50.
- MANTON, J. G., & HUGHES, P. K. (1990). Aircrew tasks and cognitive complexity. *Proceedings of the First Australian Aviation Psychology Conference*.
- MORAY, N., & DESALVO, T. (1991). *Linguistic aspects and navigational instruction under feedback versus no feedback constraints*. Manuscript in preparation.
- NISBETT, R. E., & WILSON, T. D. (1977). Telling more than we can know: Verbal reports on mental processes. *Psychological Review*, **84**, 231-259.
- POLTROCK, S. E., & NASR, M. G. (1989). *Protocol analysis: A tool for analyzing human-computer interactions* (MCC Tech. Rep. ACT-HI-186-89). Austin, TX: Microelectronics and Computer Technology Corporation.
- PRAETORIUS, N., & DUNCAN, K. (1988). Verbal reports: A problem in research design. In L. P. N. Goodstein, S. E. Olsen, & H. B. Andersen (Eds.), *Tasks, errors and mental models: A Festschrift to celebrate the 60th birthday of Jens Rasmussen* (pp. 293-314). London: Taylor & Francis.
- QUERA, V., & ESTANY, E. (1984). ANSEC: A BASIC package for lag sequential analysis of observational data. *Behavior Research Methods & Instrumentation*, **16**, 303-306.
- RAHIMI, M., & AZEVEDO, G. (1990). A task analysis of industrial robot teach programming. In W. Karwowski & M. Rahimi (Eds.), *Ergonomics of hybrid automated systems II*. Amsterdam: Elsevier Science.
- REYNOLDS, H. T. (1984). *Analysis of nominal data* (2nd ed.). Beverly Hills, CA: Sage.
- SACKETT, G. P. (1979). The lag sequential analysis of contingency and cyclicity in behavioral interaction research. In J. Osofsky (Ed.), *Handbook of infant development* (pp. 855-878). New York: Wiley.
- SANDERSON, P. M. (1990). Verbal protocol analysis in three experimental domains using SHAPA. In *Proceedings of the Human Factors Society 34th Annual Meeting*. Santa Monica, CA: Human Factors Society.
- SANDERSON, P. M., JAMES, J. M., & SEIDLER, K. S. (1989). SHAPA: An interactive software environment for protocol analysis. *Ergonomics*, **11**, 1271-1302.
- SANDERSON, P. M., JAMES, J. M., WATANABE, L., & HOLDEN, J. (1990). Human operator behaviour in complex worlds: Rendering sequential records analytically tractable. In *Proceedings of the Ninth European Annual Conference on Human Decision Making and Manual Control*. Ispra, Italy: Committee of the European Communities Joint Research Center (Ispra Establishment). [Also, Engineering Psychology Research Laboratory Tech. Rep. EPRL-90-15, Department of Mechanical and Industrial Engineering, University of Illinois at Urbana-Champaign, Urbana, IL]
- WATERMAN, D. A., & NEWELL, A. (1971). Protocol analysis as a task for artificial intelligence. *Artificial Intelligence*, **2**, 285-318.
- WATERMAN, D. A., & NEWELL, A. (1973). *PAS-II: An interactive task-free version of an automatic protocol analysis system*. Pittsburgh, PA: Department of Computer Science, Carnegie-Mellon University.
- YODER, P. J., & TAPP, J. T., SR., (1990). SATS: Sequential analysis of transcripts system. *Behavior Research Methods, Instruments, & Computers*, **22**, 339-343.

NOTES

1. The first code for this program was dubbed SHAPA, standing for "Hemi-Semi Automated Protocol Analysis," with the first two initials transposed to make it more pronounceable, and the name has persisted. This news seems to leave questioners dissatisfied, so convention has forced us to concoct alternative decodings, such as "Software to Heuristically Assist Protocol Analysis," "Super Helpful Aid to Protocol Analysis," and other horrors.

(Manuscript received October 24, 1990;
revision accepted for publication June 19, 1991.)