
ARTICLES FROM THE SCIP CONFERENCE

CAPAS 2.0: A computer tool for coding transcribed and digitally recorded verbal reports

ROBERT J. CRUTCHER

University of Dayton, Dayton, Ohio

A new computer software tool for coding and analyzing verbal report data is described. Combining and extending the capabilities of earlier verbal report coding software tools, CAPAS 2.0 enables researchers to code two different types of verbal report data: (1) verbal reports already transcribed and stored in text files and (2) verbal reports in their original digitally recorded audio format. For both types of data, individual verbal report segments are presented in random order and coded independently of other segments in accordance with a localized encoding principle. Once all reports are coded, CAPAS 2.0 converts the coded reports to a formatted file suitable for analysis by statistical packages such as SPSS.

Verbal reports can provide unique behavioral data for inferring psychological processes (see, e.g., Anderson, 1987; Chi, 1997; Crutcher 1998; Crutcher & Ericsson, 2000; Ericsson & Simon, 1993; Siegler, 1987, 1988; Simon & Kaplan, 1989). Although there have been challenges to the validity and reliability of verbal report data (see, e.g., Nisbett & Wilson, 1977), researchers have identified conditions under which reliable and valid verbal report data can be collected (Chi, 2006; Crutcher, 1994; Ericsson, 2002; Ericsson, 2006; Ericsson & Crutcher, 1991; Ericsson & Simon, 1993; Smith & Miller, 1978; Trabasso & Suh, 1993; P. White, 1980). As a result, verbal reports are now used across diverse task domains (Austin & Mawhinney, 1999; Crutcher, 1998; Crutcher & Ericsson, 2000; Ericsson, 2006; Ericsson, Charness, Feltovich, & Hoffman, 2006; Ericsson & Simon, 1993; Siegler, 1987, 1988; Simon & Kaplan, 1989; Trabasso & Suh, 1993) and have proved useful in testing competing theoretical claims (see, e.g., Trabasso & Suh, 1993). Nonetheless, important practical and theoretical concerns remain, to which researchers using verbal reports must attend. Proper task analysis prior to collecting reports, as well as the use of proper methods to elicit reports (e.g., asking participants to simply report their thoughts while doing a task and not to explain or justify specific behaviors), are examples of these concerns (see, e.g., Chi, 2006; Crutcher, 1994; Crutcher, 2003; Crutcher, Ericsson, & Wichura, 1994; Ericsson, 2006; Schraagen, 2006).

Another important concern of those who use verbal report data in their research is the amount of data that must be processed and analyzed. Researchers have stressed the importance of computer tools to make the process more

efficient and objective (Ames et al., 2005; Bainbridge & Sanderson, 1995; Crutcher, 2003; Crutcher et al., 1994; Ericsson & Simon, 1993; Fisher, 1987; James & Sanderson, 1991; MacLin & MacLin, 2005; Sanderson, James, & Seidler, 1989). My focus in this report is on the use of one such tool to improve the coding of verbal report data. Specifically, I describe a new computer tool, CAPAS 2.0¹ (Computer-Aided Protocol Analysis System), that can be used to code verbal report data of two different types: verbal reports that are in text form—such as verbal reports that have been transcribed—and verbal reports that are still in their original recorded format. I first provide a brief background on the use of computer systems in coding and analyzing verbal report data. I then describe the design philosophy of CAPAS 2.0 and how the various parts of the program work. I conclude by comparing CAPAS 2.0 to other computer coding systems and offer some suggestions for future development of the program.

Computer systems for coding verbal reports may be classified into two general types. In automated or automatic computerized coding systems, the coding of the verbal reports is accomplished entirely by a computer program (Waterman & Newell, 1971, 1973), whereas in semiautomated or semiautomatic computer coding systems, a computer program assists human coders in coding verbal reports (Bainbridge & Sanderson, 1995; Bhaskar & Simon, 1977; Crutcher, 2003; Crutcher & Ericsson, 2000; Crutcher et al., 1994; Ericsson & Simon, 1993; Fisher, 1987; James & Sanderson, 1991; Sanderson, James, & Seidler, 1989). The PAS-I system of Waterman and Newell is an example of the first type of system. PAS-I works by first defining a problem space to limit the number

R. J. Crutcher, crutcher@udayton.edu

of coding categories onto which the verbal reports are mapped. The verbal reports are then segmented manually, processed by a linguistic parser, mapped onto a set of propositional relations, and reduced to a smaller set of semantic primitives. The primitives are translated into a problem behavior graph that captures a participant's successive knowledge states and the productions necessary to move from one state to the next. A more recent example of an automated coding system is the one developed by D. J. White, King, and Duncan (2002), in which the researchers trained a computer system to recognize a custom vocabulary of approximately 200 words designed to classify bird behaviors. This was a spoken vocabulary, used by the researchers while observing the birds in their natural environment; the verbal report responses were automatically processed and coded by the computer program. The advantage of an automatic coding approach is that all of the knowledge and coding rules are explicitly defined and incorporated into the program, ensuring that all protocol segments are coded objectively and consistently using the actual coding rules. The disadvantage of this approach is the difficulty of writing such programs. Such systems are generally only practical in well-defined task domains in which the vocabulary or range of utterances is limited.

Semiautomated coding systems, on the other hand, are computer tools that aid human coders in coding verbal reports (Ames et al., 2005; Bainbridge & Sanderson, 1995; Bhaskar & Simon, 1977; Crutcher, 2003; Crutcher & Ericsson, 2000; Crutcher et al., 1994; Ericsson & Simon, 1993; Fisher, 1987; James & Sanderson, 1991; MacLin & MacLin, 2005; Sanderson et al., 1989). These programs can increase the objectivity, efficiency, and ease of coding verbal reports by improving how protocol segments are presented to human coders and by providing support in applying coding rules consistently across a large set of verbal reports. Semiautomated coding systems divide responsibility for coding decisions between the program and the coder, with individual systems varying as to the relative responsibilities of the program and the coder. One example of a semiautomated coding system in which considerable domain knowledge is contained in the program is SAPA (Bhaskar & Simon, 1977). SAPA models a participant's problem-solving strategies and can predict the next operation the participant will apply; however, the human coder has the option of changing the program's decision and returning the program to the correct path.

SAPA is an exception, since most semiautomated systems contain little or no specific domain knowledge in the system per se, although the researcher can specify rules or codes when setting up the system for a specific coding project. There are many such coding systems (Ames et al., 2005; Bainbridge & Sanderson, 1995; Crutcher, 2003; Crutcher et al., 1994; Ericsson & Simon, 1993; Fisher, 1987; James & Sanderson, 1991; MacLin & MacLin, 2005; Sanderson, James, & Seidler, 1989). These systems assist the coder by handling large numbers of verbal reports while focusing the coder's attention on one or a few reports at a time. The advantage of semiautomated systems is applicability: Because they do not require hard coding of domain-specific knowledge, many of these systems

can be used in a variety of research areas. Nonetheless, such semiautomated systems, depending on how they are implemented, do have the potential for bias, because they leave coding decisions to human coders. For example, the degree of reliability in applying rules for many of these systems is determined by factors controlled not by the computer system, but by the researcher, factors such as the clarity of the coding rules and how well coders are trained (MacLin & MacLin, 2005). Furthermore, as previous researchers have emphasized, a thorough task analysis prior to constructing a coding scheme is very important (Crutcher, 1994; Crutcher, 2003; Ericsson, 2006; Ericsson & Simon, 1993; Schraagen, 2006).

Another potential disadvantage of semiautomatic coding systems is that coding bias cannot be entirely eliminated. Whereas a completely automated system codes the same segment the same way regardless of context, a human coder cannot help but take account of the context in which a given verbal report is coded. For example, information from previously coded segments for the same participant or knowledge of the specific experimental condition in which a trial occurred may influence the coding of the current segment. One potential solution to this dilemma is to define independent verbal reports or report segments for data from an entire group of participants and present these segments randomly to coders without contextual information such as the condition or trial number associated with a report. This approach of minimizing the context surrounding a report that is being coded is referred to as the *localized coding* (or *encoding*) *principle* (Crutcher, 1994, 2003; Ericsson & Simon, 1993), and it is an important design feature of the system described here.

General Design Philosophy and Overview of CAPAS 2.0

Many systems for coding verbal report data, as well as a wealth of other behavioral data, are available. As far as I can determine, however, none of these systems specifically tries to implement a localized coding approach. According to the localized coding principle, a verbal report (or a segment of a report—e.g., a phrase or a sentence) should be coded with as little reliance on surrounding context as possible. When a coder classifies a verbal report or a segment from a report, that classification should be based on a classification rule only, and information that might influence the coder's decision, such as the specific trial, condition number, or participant, should be minimized as much as possible. The intent of this principle is to apply the same independence assumption to verbal report data that is routinely applied to other types of behavioral data (see Ericsson & Simon, 1993, for a thorough discussion). The earlier MPAS (Crutcher et al., 1994; Ericsson & Simon, 1992) and CAPAS 1.0 systems (Crutcher, 2003) both embodied this principle in their designs.

CAPAS 2.0 is a new coding system whose design principles reflect those of the earlier MPAS (Crutcher et al., 1994) and CAPAS programs (Crutcher, 2003). CAPAS 2.0 combines the most important capabilities of the MPAS and CAPAS programs while extending their functionality. CAPAS 2.0 can be used to code text-based verbal proto-

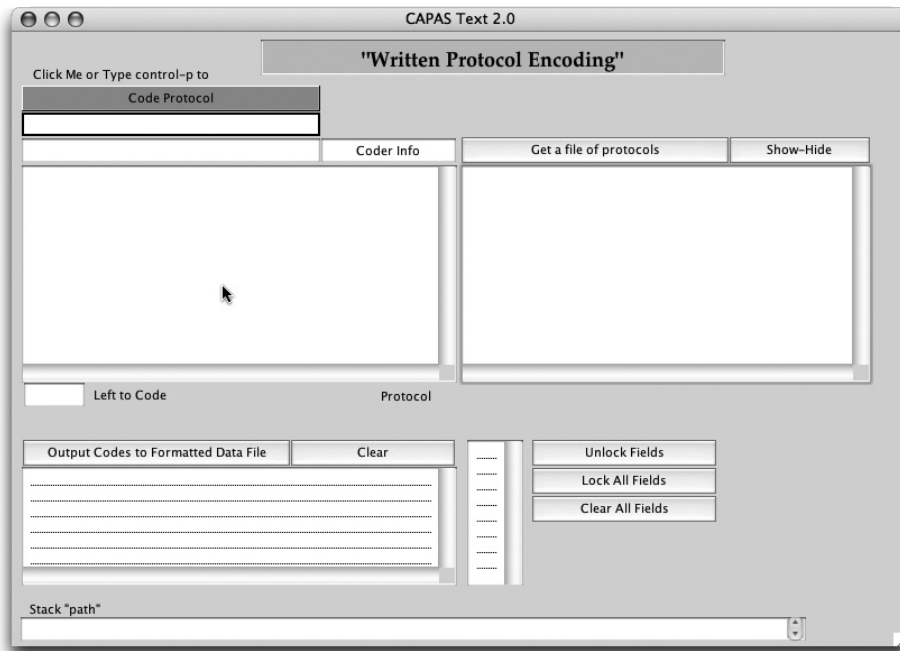


Figure 1. CAPAS 2.0 text module interface, upon starting the program.

cols as well as verbal protocols that are digitally recorded and are still in the original digital format. For both types of verbal reports, CAPAS stores all of the verbal reports for a given study together and presents them one at a time in a random sequence. I will first describe the text-based part of the program, which codes transcribed verbal reports, then the audio verbal report part, and finally a coding helper tool.

Using CAPAS 2.0 to Code Text-Based Verbal Reports

CAPAS 2.0 can be used to code text-based verbal reports, such as reports that originally were stored on audio tape and subsequently transcribed. However, before CAPAS 2.0 can operate on these verbal reports, they must be formatted and stored all together in a text file. The following is an example of the specific formatting required for an individual verbal report:

```
9 1 1 -1 2 -1 27 11 *candada candle padlock*
candada
candle
same can sounds hard c
candle padlock
a padlock used as a base for a candle\
```

Individual verbal reports are demarcated by the backslash character “\” placed at the end of the protocol. Preceding the text of each protocol, on the first line, an optional header may also be placed. This header may include information (such as the subject, condition, and trial number) that is invisible to the coder during coding but will be stored with the codes for the verbal reports for later output and analysis. Information that the researcher wants the coder to see while coding can be identified by placing

the “*” character before and after the information. During coding, the part of the header surrounded by asterisks will be visible to the coder, whereas the remaining header information will remain invisible. After all verbal reports are coded, the header information and the codes for the reports are written to an output file in a form suitable for analysis by a statistical package such as SPSS.

Once the reports are properly formatted and stored in a text file, the coder opens CAPAS Text 2.0 to begin coding (see Figure 1). CAPAS 2.0 first checks the file to see that it is properly formatted and then displays it in a scrolling window so that the coder can look for problems if necessary. This window disappears as soon as coding begins. To begin the actual coding of reports, the user clicks the Code Protocol button, and a randomly selected protocol appears in a scrolling window display. If there is any information that the coder should see for that particular verbal report, it appears in a field just above the display window for the verbal report. The coder types a 5-digit code and hits the return key to enter the code and proceed to the next report. The coder can use either the graphic display buttons or keyboard shortcuts to select and present the verbal reports. Throughout coding, CAPAS 2.0 shows how many reports have been coded and how many are left to code.

Since CAPAS 2.0 is designed to work with large files of verbal reports, all stored in a single file, it allows the coder to quit at any time and resume coding later. In addition, CAPAS 2.0 is designed to forestall any loss of data in the event of problems, such as an unexpected crash of the program. When CAPAS 2.0 first reads a file of verbal reports for coding, it copies the reports to a new file in which the verbal reports and their associated codes will be stored. If for some reason the program crashes or malfunctions, the file containing the verbal reports and codes can

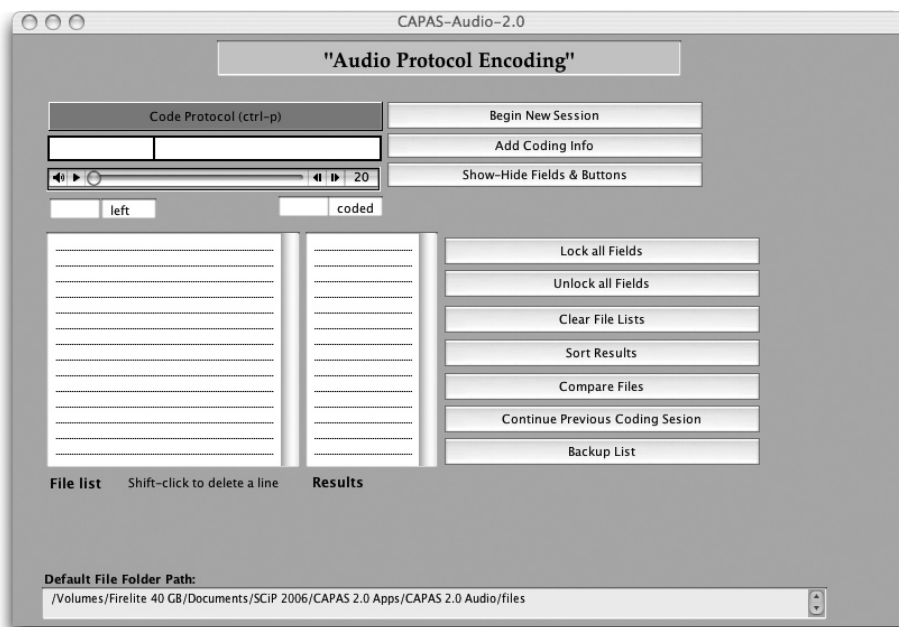


Figure 2. CAPAS 2.0 audio module interface, upon starting the program.

be read back into the program, and coding can continue from where the program left off. When all verbal reports have been coded, CAPAS 2.0 alerts the user. CAPAS 2.0 can then generate an output file with complete header information and codes in a form suitable for analysis by a statistical package such as SPSS.

Using CAPAS 2.0 to Code Verbal Reports in Audio Form

Although the text-based coding part of CAPAS 2.0 is useful for coding a large number of verbal reports, the downside is that the reports must be transcribed from audio recordings. The transcription process is not only the most time-consuming part of the verbal report processing and analysis, but it can also introduce errors into the data. A transcriber may mishear what was said on part of the recording and transcribe the wrong phrase or may hear correctly but still make errors transcribing. CAPAS 2.0 allows the researcher to code directly from the original audio recordings of the verbal reports, eliminating the need for transcribing the protocols. It does this while still enforcing the localized coding principle.

Using CAPAS 2.0 to code verbal protocols in audio form is straightforward and similar to coding text-based reports; however, before coding can begin, the reports must be pre-processed, and this preprocessing is different than that for text-based reports. Whereas the text module of CAPAS 2.0 requires that all verbal reports be stored in a single text file, the audio module requires that all of the individual verbal reports or report segments be stored in individual sound files (e.g., AIFF or WAV files) and that the files be named in a way that allows CAPAS 2.0 to process them. Each audio file must be given a filename that consists of a series of hyphenated numbers (e.g., 1-1, 1-2, 1-3 . . . 2-1, 2-2, 2-3,

etc.). This naming system allows CAPAS 2.0 to randomize the presentation of the audio segments yet maintain the identity of the segments throughout coding and, at the end, sort out which code goes with which protocol. The researcher must therefore define a numbering scheme such that each filename uniquely identifies a specific protocol for a specific trial and participant (e.g., 3-11 would be the verbal report for the 11th trial for participant number 3).

Putting the audio recordings into the required format is not difficult to accomplish. The simplest and most direct method is to use computer audio software to digitally record and segment the verbal reports automatically while running a subject. Bias's Peak² software, for example, can be used to record the verbal protocols while running a subject in an experiment. While recording, Peak allows marking of audio segment boundaries or regions with a simple keystroke operation. Later, these keystroke-defined regions are easily exported to individual sound files with a single batch command. Furthermore, Peak allows these individually marked and exported sound regions to be named automatically in accordance with a preestablished prefix-naming scheme, so that, for example, the first segment would be named 1-1, the next 1-2, and so forth. With this approach, the verbal protocols are already segmented and ready for processing immediately after a subject has been run. It is also possible, with a digital recording, to accomplish the marking and exporting operation after the fact; thus in principle, this approach could be employed with analogue recordings if they were first digitized. However, this would entail considerably more work, time, and effort than the first approach.

All of the audio files for all participants in an experiment are placed together in a file folder with the CAPAS 2.0 application. Additional information that the researcher

wants to present to the coder during coding is placed in a separate coding information file, which must be organized so that the coding information associated with each protocol is located on a single line of the file. Furthermore, the order of lines must correspond with the order of the verbal report file-numbering scheme just described. In practice, this is straightforward if the protocols are collected in a laboratory experiment in which participants proceed through a series of trials and Peak is used to generate the audio files for those trials (as previously described); the order of the audio file names will correspond to the trial sequence and will thus be aligned with any information in the experiment's output file. Editing this file down to whatever information (e.g., the stimulus presented) should be shown for a given trial during the coding of the reports is an easy matter.

When the audio coding module of CAPAS 2.0 starts up, the user is greeted by a graphical interface similar to the one for the text module (see Figure 2). When initiating a new coding session, CAPAS 2.0 generates a master index of all the audio filenames stored in the files folder. This index is used to randomly select and play the audio files. As each verbal report is coded, the code, along with the identifying number of the verbal report, is written to a results file. CAPAS 2.0 keeps track of how many reports have been coded and how many remain to be coded. Coding can stop at any point and resume later where the program left off.

CAPAS 2.0 presents each verbal report to the coder by playing the corresponding audio file for that report. The coder clicks on the Code Protocol button or types a keyboard shortcut (control-p) to play a report, types the code into a preselected field, and hits the return key to enter the code. Only a single code may be entered, and this code must consist of a string of no more than 5 numbers (e.g., 1, 1, 1, 0, 1). In contrast to CAPAS Version 1.0, in which an audio segment was played in its entirety and could not

be interrupted or replayed by the user, a graphic controller in CAPAS 2.0 (see Figure 2) allows the coder to replay the entire verbal report or a specific part of the report as many times as necessary. CAPAS 2.0 alerts the user once all verbal reports have been coded, at which point it can generate an output file containing the codes, subject, and trial information in a form suitable for analysis by a statistical package such as SPSS.

CAPAS 2.0 Code Helper

In addition to the text and audio coding modules, CAPAS 2.0 has another tool to aid the coder in applying classification rules. CAPAS 2.0 expects input consisting of numeric strings up to 5 digits long. It is up to the researcher to define a coding system for these numbers. Each digit of the 5-digit string will likely code a different variable for a given report. CAPAS 2.0 provides a tool that can help coders, especially novice coders, in mapping the coding categories to the numeric codes. This tool (see Figure 3) displays the rules or categories represented by each of the 5 numbers entered and can remain on the screen while the coder is using either of the other modules.

To use the tool, the coder simply clicks on the relevant rule, and the numeric code associated with that rule or category is written to the corresponding position of a 5-digit string. This operation can be repeated for each of the 5 numbers to be input, at which point a button click transfers the completed 5-digit code to the entry field of the CAPAS 2.0 text or audio module. Although this method is less efficient than typing codes directly into the entry field of the text or audio module, it can save considerable time for a novice coder, who would otherwise have to read and translate codes from a printed rule sheet and then type them into the code entry field. As a coder becomes more proficient, the helper may be placed next to the module while the coder works directly in the module, using the

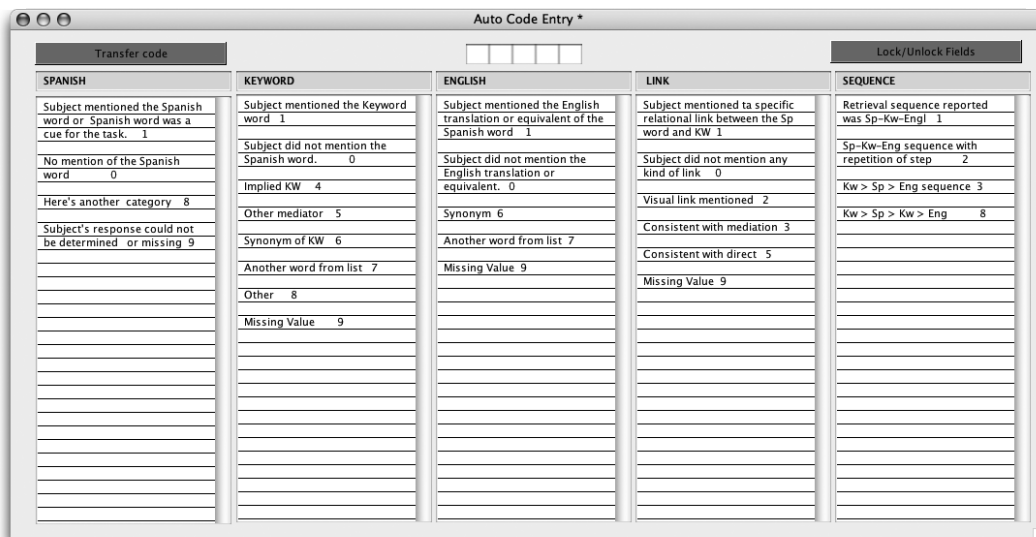


Figure 3. CAPAS 2.0 coding helper interface.

coding helper merely to confirm, when necessary, what is typed directly into the code entry field of the audio or text module. The rules and numbers used by the coding helper can be easily changed to suit the needs of a particular coding scheme.

A Comparison of CAPAS 2.0 With Other Programs

A variety of behavioral coding systems are available these days. Many of these systems can handle a wider range of behavioral data than CAPAS 2.0 (see, e.g., James & Sanderson, 1991; MacLin & MacLin, 2005; Noldus, Trienes, Hendriksen, Jansen, & Jansen, 2000), and some of them have rich environments for coding a variety of responses in different ways. CAPAS 2.0, on the other hand, focuses on coding verbal report data, and the specific way in which CAPAS 2.0 handles such data is different from most of these other systems. The Observational Data Coding System (ODCS; MacLin & MacLin, 2005) provides a powerful tool for coding social interactions and to that end provides tools that allow the coder to play an audio or video record of a conversation, for example, and code many different types of behavioral data. It also permits the coder to define a range of coding categories and subcategories that are later displayed as button options on which the coder can click to classify responses easily. CAPAS 2.0 is not nearly as sophisticated or rich in terms of interface or capability as programs like ODCS. However, these programs do not permit randomized presentation of individual segments or protocols without surrounding context. CAPAS 2.0's strength is that it forces the researcher to presegment a large set of verbal reports in a theoretically neutral way (e.g., by trials, if the reports are brief reports such as one might employ in laboratory paradigms, or by speech pauses if the reports are long) and then present these segments randomly without the surrounding context. In other words, when coding a report, the coder is blind to the condition, the trial, and the part of the experiment whence a given report comes.

Thus CAPAS 2.0 embodies in its design the localized encoding principle (Crutcher, 1994; Crutcher 2003; Crutcher et al., 1994; Ericsson & Simon, 1993). As Ericsson and Simon have emphasized, it is important to address concerns about possible bias in the treatment of verbal report data in order to place verbal report data on par with other types of behavioral data. Of course, a localized coding approach may not always be the best approach to handling verbal report data. In conversational analysis research, for example, the researcher may want to view and play surrounding context (e.g., what preceded a conversational interaction) and replay conversational transactions to code significant aspects of exchanges. However, in other paradigms, especially in experimental, laboratory-based paradigms, it is important that the coder not know which condition or trial a given verbal report is associated with. For such paradigms, CAPAS 2.0 is extremely useful in that it limits the potential for coder bias.

CAPAS 2.0 is one in a line of verbal report coding programs that have implemented the localized encoding principle in their design. The earlier MPAS program (Crutcher

et al., 1994) and an earlier version of CAPAS (Crutcher, 2003) were designed with similar principles in mind. However, CAPAS 2.0 represents a significant improvement over these earlier programs in several ways. First, CAPAS 2.0 combines the capabilities of the MPAS and CAPAS programs into a single unified program environment that can be used to code text-based verbal reports or audio recordings of reports. Second, whereas MPAS interacted with the user by means of a command line interface, CAPAS 2.0 employs a graphical interface that is easier to interact with. CAPAS 2.0 is flexible: It permits codes to be entered completely from the keyboard, as does MPAS, or by using the code entry helper with select and click operations. This flexibility is particularly important when a new coding scheme is adopted or when a coder who is unfamiliar with the coding categories of a particular study uses the system. A test comparing the old MPAS program with the current CAPAS 2.0 program was conducted using a novice coder to code 20 verbal reports from an earlier study. For the MPAS test, the coder had a written version of the coding rules to consult while coding the reports; for the CAPAS 2.0 test, the coder consulted and used the CAPAS 2.0 onscreen helper while entering the codes. It took less time to code the reports using CAPAS 2.0 (5 min 21 sec) than it took using the old MPAS program to code a similar set of 20 reports (7 min 43 sec). This represents an approximately 30% savings in time.

The method of presenting the audio files is much improved in the CAPAS 2.0 audio module as compared with the earlier version of CAPAS (Crutcher, 2003). Instead of only listening to the recording, the coder can now see the recording being played—that is, the sound file is presented using a QuickTime Media player (see Figure 2). This should improve coding reliability and validity because if something is not heard clearly, the verbal report or the relevant section of the report can be easily replayed as many times as necessary. In the older version of the program, audio segments were played in their entirety, with no control over starting, stopping, or choosing to play a specific part of the recording. Of course, the biggest advantage of using CAPAS 2.0 to code the verbal reports directly from the audio recordings is that the transcription process is eliminated. An earlier test of CAPAS (Crutcher, 2003) found that this effectively reduced the time necessary to process and code a set of verbal reports by more than 60% (Crutcher, 2003) compared with a text-based coding program such as MPAS.

Summary, Limitations, and Future Development

As a tool to improve the efficiency of coding verbal protocol data (text-based verbal reports, previously transcribed reports, or verbal reports still in the raw recorded format), CAPAS 2.0 should prove quite useful. Unlike other coding tools, CAPAS 2.0 can code a large number of verbal reports or report segments independently, while adhering to a localized coding principle. The current version of CAPAS is significantly improved over the earlier version of the CAPAS program (Crutcher, 2003), as well as the program's predecessor, MPAS (Crutcher et al., 1994).

Like any computer tool, CAPAS 2.0 has limitations. The coding helper described in this report, although a significant improvement over the earlier versions of the program in which the coder had to consult an external document for the coding rules and categories, is not nearly as flexible as it could be. It needs to be rewritten to work more seamlessly with the other two modules of the program. Ideally, the coding helper would, when needed, be accessible from directly within the module used to code the protocols and otherwise would stay out of the way.

Although CAPAS 2.0 has clear advantages over programs that require first transcribing verbal protocols, working directly with audio files presents potential problems that should be acknowledged. First, coding protocols directly from audio recordings is probably practical only when the protocols are of brief duration, as in retrospective reports gathered in memory or strategy experiments (see, e.g., Crutcher & Ericsson, 2000; Siegler, 1987, 1988). Longer protocols may exceed the capacity of a coder to process and code accurately. In the new version of CAPAS, this is less of a problem since audio recordings can be easily replayed. However, a coder may still find it difficult to listen to and process a very long recording. The advantage of transcribed text is that the coder can easily scan back over the report to find a salient piece of information. In the case of a recording, the coder must skip back over the audio in a less directly guided manner. It is possible, then, that verbal reports of relatively brief duration may be closer to the ideal for direct coding of audio recordings. Empirical studies could test this. This limitation, if it is one, is not just of CAPAS 2.0 but of any system used to code audio recordings of verbal protocols.

Another concern that is not a limitation of CAPAS per se, but of any program that codes multiple verbal reports directly from audio recordings, is that audio recordings may not be ideal if the coding approach wishes to minimize contextual information during coding. An audio recording may provide contextual information that is absent from a transcribed report. During the coding of a written verbal report, all of the contextual information as to subject, trial, condition, and so forth may be hidden from the coder. However, when listening to reports, coders may be able to identify the speaker and to relate a current coding to one previously heard from the same speaker. Whether this is a concern likely depends on the number of reports and participants being analyzed. In a large pool of reports, with many different speakers, it is much less likely that the coder will remember any specific information from a previous speaker's reports. However, in studies involving only a couple of speakers, the possibility that a coder may identify a speaker may be cause for concern. Again, this problem is not unique to CAPAS 2.0. However, it is one argument for not abandoning transcription of verbal reports in all cases. This is one of the reasons that CAPAS 2.0 was designed to handle transcribed verbal reports as well as audio recordings.

In conclusion, CAPAS 2.0 is a useful new tool for coding and analyzing verbal protocol data in domains in which the independent coding of protocols is important to the researcher, most likely in laboratory experiments.

The current version of CAPAS 2.0 improves on previous programs such as MPAS by reducing the time necessary to process and code protocol data as well as by making the coding of verbal reports easier. Finally, CAPAS 2.0 enforces a localized coding principle to ensure that the verbal report data are coded and analyzed as objectively as possible.

AUTHOR NOTE

Inquiries regarding the availability of the CAPAS 2.0 program, and other correspondence, should be addressed to R. J. Crutcher, Department of Psychology, University of Dayton, 300 College Park, Dayton, OH 45469-1430 (e-mail: crutcher@udayton.edu).

REFERENCES

- AMES, S. L., GALLAHER, P. E., SUN, P., PEARCE, S., ZOGG, J. B., HOUSKA, B. R., ET AL. (2005). A Web-based program for coding open-ended response protocols. *Behavior Research Methods*, *37*, 470-479.
- ANDERSON, J. R. (1987). Methodologies for studying human knowledge. *Behavioral & Brain Sciences*, *10*, 467-505.
- AUSTIN, J., & MAWHINNEY, T. C. (1999). Using concurrent verbal reports to examine data analyst verbal behavior. *Journal of Organizational Behavior Management*, *18*, 61-81.
- BAINBRIDGE, L., & SANDERSON, P. (1995). Verbal protocol analysis. In J. R. Wilson & E. N. Corlett (Eds.), *Evaluation of human work: A practical ergonomics methodology*. Bristol, PA: Taylor & Francis.
- BHASKAR, R., & SIMON, H. A. (1977). Problem solving in semantically rich domains: An example from engineering thermodynamics. *Cognitive Science*, *1*, 193-215.
- CHI, M. T. H. (1997). Quantifying qualitative analyses of verbal data: A practical guide. *Journal of the Learning Sciences*, *6*, 271-315.
- CHI, M. T. H. (2006). Laboratory methods for assessing experts' and novices' knowledge. In K. A. Ericsson, N. Charness, P. J. Feltovich, & R. R. Hoffman (Eds.), *The Cambridge handbook of expertise and expert performance* (pp. 167-184). New York: Cambridge University Press.
- CRUTCHER, R. J. (1994). Telling what we know: The use of verbal report methodologies in psychological research. *Psychological Science*, *5*, 241-244.
- CRUTCHER, R. J. (1998). The role of prior knowledge in mediating foreign vocabulary acquisition and retention: A process-analytic approach. In A. F. Healy & L. E. Bourne, Jr. (Eds.), *Foreign language learning: Psycholinguistic studies on training and retention* (pp. 91-111). Mahwah, NJ: Erlbaum.
- CRUTCHER, R. J. (2003). A computer-aided digital audio recording and encoding system for improving the encoding of verbal reports. *Behavior Research Methods, Instruments, & Computers*, *35*, 263-268.
- CRUTCHER, R. J., & ERICSSON, K. A. (2000). The role of mediators in memory retrieval as a function of practice: Controlled mediation to direct access. *Journal of Experimental Psychology: Learning, Memory, & Cognition*, *26*, 1297-1317.
- CRUTCHER, R. J., ERICSSON, K. A., & WICHURA, C. A. (1994). Improving the encoding of verbal reports by using MPAS: A computer-aided encoding system. *Behavior Research Methods, Instruments, & Computers*, *26*, 167-171.
- ERICSSON, K. A. (2002). Towards a procedure for eliciting verbal expression of non-verbal experience without reactivity: Interpreting the verbal overshadowing effect within the theoretical framework for protocol analysis. *Applied Cognitive Psychology*, *16*, 981-987.
- ERICSSON, K. A. (2006). Protocol analysis and expert thought: Concurrent verbalizations of thinking during experts' performance on representative tasks. In K. A. Ericsson, N. Charness, P. J. Feltovich, & R. R. Hoffman (Eds.), *The Cambridge handbook of expertise and expert performance* (pp. 223-241). New York: Cambridge University Press.
- ERICSSON, K. A., CHARNNESS, N., FELTOVICH, P. J., & HOFFMAN, R. R. (Eds.). (2006). *The Cambridge handbook of expertise and expert performance*. New York: Cambridge University Press.
- ERICSSON, K. A., & CRUTCHER, R. J. (1991). Introspection and verbal reports on cognitive processes: Two approaches to the study of thinking. *New Ideas in Psychology*, *9*, 57-71.

- ERICSSON, K. A., & SIMON, H. A. (1993). *Protocol analysis: Verbal reports as data* (Rev. ed.). Cambridge, MA: MIT Press.
- FISHER, C. (1987). Advancing the study of programming with computer-aided protocol analysis. In G. M. Olson, S. Sheppard, & E. Soloway (Eds.), *Empirical studies of programmers: Second workshop* (pp. 198-216). Norwood, NJ: Ablex.
- JAMES, J. M., & SANDERSON, P. M. (1991). Heuristic and statistical support for protocol analysis with SHAPA Version 2.01. *Behavior Research Methods, Instruments, & Computers*, **23**, 449-460.
- MACLIN, O. H., & MACLIN, M. K. (2005). Coding observational data: A software solution. *Behavior Research Methods*, **37**, 224-231.
- NISBETT, R. E., & WILSON, T. D. (1977). Telling more than we can know: Verbal reports on mental processes. *Psychological Review*, **84**, 231-259.
- NOLDUS, L. P. J. J., TRIENES, R. J. H., HENDRIKSEN, A. H. M., JANSEN, H., & JANSEN, R. G. (2000). The Observer Video-Pro: New software for the collection, management, and presentation of time-structured data from videotapes and digital media files. *Behavior Research Methods, Instruments, & Computers*, **32**, 197-206.
- SANDERSON, P. M., JAMES, J. M., & SEIDLER, K. S. (1989). SHAPA: An interactive software environment for protocol analysis. *Ergonomics*, **32**, 1271-1302.
- SCHRAAGEN, J. M. (2006). Task analysis. In K. A. Ericsson, N. Charness, P. J. Feltovich, & R. R. Hoffman (Eds.), *The Cambridge handbook of expertise and expert performance* (pp. 185-201). New York: Cambridge University Press.
- SIEGLER, R. S. (1987). The perils of averaging data over strategies. *Journal of Experimental Psychology: General*, **116**, 250-264.
- SIEGLER, R. S. (1988). Strategy choice procedures and the development of multiplication skill. *Journal of Experimental Psychology: General*, **117**, 258-275.
- SIMON, H. A., & KAPLAN, C. A. (1989). Foundations of cognitive science. In M. I. Posner (Ed.), *Foundations of cognitive science* (pp. 1-47). Cambridge, MA: MIT Press.
- SMITH, E. R., & MILLER, F. D. (1978). Limits on perception of cognitive processes: A reply to Nisbett and Wilson. *Psychological Review*, **85**, 355-362.
- TRABASSO, T., & SUH, S. (1993). Understanding text: Achieving explanatory coherence through on-line inferences and mental operations in working memory. *Discourse Processes*, **16**, 3-34.
- 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. In *Proceedings of the Third International Joint Conference on Artificial Intelligence* (pp. 431-445). Menlo Park, CA: Stanford Research Institute.
- WHITE, D. J., KING, A. P., & DUNCAN, S. D. (2002). Voice recognition technology as a tool for behavioral research. *Behavior Research Methods, Instruments, & Computers*, **34**, 1-5.
- WHITE, P. (1980). Limitations on verbal reports of internal events: A refutation of Nisbett and Wilson and of Bem. *Psychological Review*, **87**, 105-112.

NOTES

1. CAPAS 2.0 can be run with a free Player application that is available at the Revolution software Web site (www.runrev.com). CAPAS 2.0 was developed using Revolution 2.7.4. Revolution is a cross-platform programming environment designed for developing and running multimedia programs on multiple operating systems, including all versions of Windows and Mac OS as well as many versions of UNIX. Although CAPAS 2.0 was written to run under Mac OS X, it should, with minor modifications, run under any version of Windows or UNIX. If the user has the Revolution Development Software, CAPAS 2.0 can be easily modified to suit the specific user's needs. Further information about CAPAS 2.0 and copies of it are available upon request from the author. A stand-alone version of CAPAS 2.0 for Mac OS X is also available. Stand-alone versions of the program for Windows XP and Mac OS 9 may be available in the future.

2. On the Apple Macintosh platform, I have found Bias's Peak program to be the best digital audio recording and editing program because of its advanced features for marking sound regions and automating the exporting of these regions to individual sound files. On the Windows platform, there are many options as well (e.g., Sound Forge and Cubase VST), but I am not familiar enough with them to recommend one over another.

(Manuscript received December 8, 2006;
revision accepted for publication January 10, 2007.)