Visualization of Computer Users' Cognitive Strategies and Behaviors

John B. Smith, Dana Kay Smith, Eileen Kupstas, and Peter Calingaert

Department of Computer Science, University of North Carolina at Chapel Hill, NC 27599-3175, U.S.A.

INTRODUCTION

Our research focuses on supporting scientists and engineers working collaboratively on shared artifacts such as documents, images, and programs. To support them fully, we need to understand their cognitive activities as they interact both with the shared artifact and with each other. Our future studies of the collective cognition of groups will be guided by our earlier work in studying the cognitive activities of a single individual. Focusing initially on the task of expository writing, we have designed a computer system to support writers, developed a theory of cognitive processing, and implemented several tools to help us visualize the writers' cognitive processes.

We began with two major goals: first, to develop an advanced writing environment suitable for professionals who write as a part of their jobs and, second, to use that system to study writers' cognitive strategies. A major theoretical component was a theory of cognitive modes and the strategies that guide writers in moving among these modes. This theory was synthesized from research in composition theory, cognitive psychology, and our own experiences as writers.

Our hypertext-based writing system [1] is multimodal: individual windows on the screen support either one or two specific cognitive modes used by writers. Each window is thus specialized to help the writer accomplish a specific portion of the over-all writing task. The system also permits objects, such as a cluster of related ideas, to be moved from one mode to another for further work. In Figure 1, the four system modes can be seen in the default screen layout (each can be expanded to cover the entire screen). The upper left mode is used for exploration, the lower left for organizing and global editing, the lower right for writing and sentence editing, and the upper right for coherence editing.

CLASSICAL TECHNIQUES

A critical problem for studying cognitive behavior of any kind is gaining access to valid and sufficient data. Since human thought processes cannot be observed directly, cognitive psychologists and others studying human intellectual behavior have developed several methods for observing them indirectly.

Think-Aloud Protocols

One such method, called *concurrent think-aloud protocols* [2], have provided a rich source of information for cognitive psychologists and for those studying human-computer interaction. The goal of this method is to produce a written record of subjects' trains of thought based on the subjects' own verbalization of their thinking as it occurs while they perform the task being investigated. Tasks that have been studied using this method include writing documents, solving arithmetic problems, assembling physical devices, and playing chess.

These data have provided rich materials for studying a number of mental skills. However, they pose significant theoretical, interpretive, and practical problems.

Keystroke Protocols

For studies that involve computers, an alternative form of protocol data is sometimes available in the form of a record of each key pressed by a subject. This approach is attractive for several reasons. Protocols may be recorded passively, unlike think-aloud methods. They can often be recorded by the operating system or by an analytic shell without modifying the application program. However, keystroke data are very fine-grained; thus, storage, management, and interpretation can be formidable tasks. To infer what the keystrokes add up to in terms of the commands and functions provided by the application program requires an analytic program analogous to the application program's own internal user interface parser that interprets user commands. For systems that use graphic displays and mouse control, many user actions will not be recorded as keystrokes; systems differ widely in their capabilities to record direct manipulation events. For these systems, keystroke data will miss many, perhaps the majority, of the user's actions. Thus, although they solve the problem posed

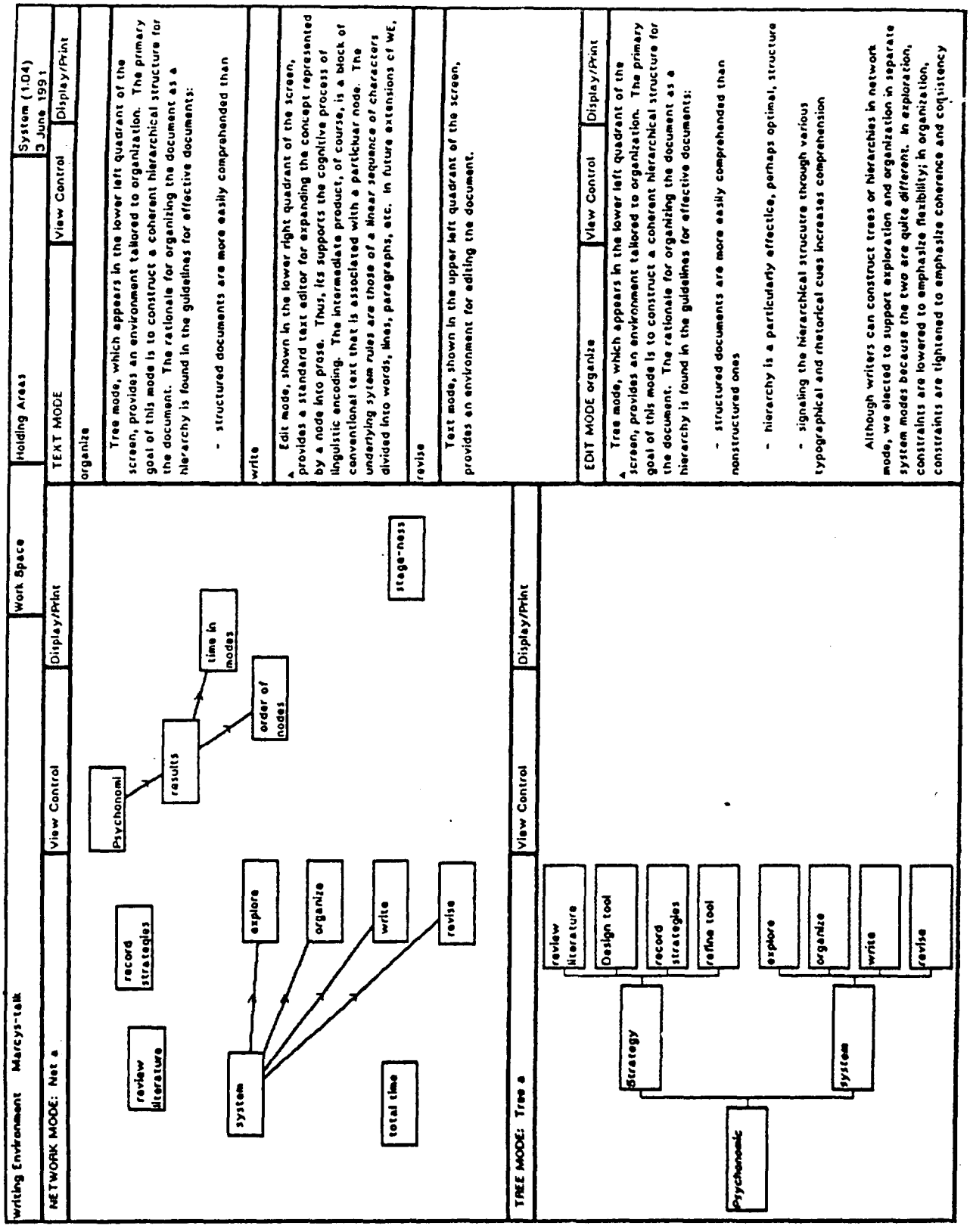

Figure 1 Writing Environment: Four System Modes

by think-aloud protocols of distorting task performance, they share the problems of incompleteness and producing large volumes of fine-grained data that must be analyzed or coded before being used.

Recordings

A third method that has been used alone as well as in conjunction with both think-aloud and keystroke protocols is the use of video and/or audio recordings. These data can show what a person is doing when not thinking aloud and not typing on the computer keyboard. They can also show what is being displayed on the computer screen – information not likely to be available through the keystroke record. Although this method produces a very rich record of behavior, it is time-intensive and requires special training and controls for the human judges who code the protocols to produce reliable, consistent data.

OUR APPROACH

We have developed an alternative approach that addresses many of these issues, but it is restricted to studies that involve use of computer systems. Our approach is to record protocols at the level of users' *actions*, rather than keystrokes. Since we study users' strategies for systems developed by our group, we are able to embed sensors directly into the programs to record movements of the mouse from one window on the screen to another, selection of objects and menu options, and character strings typed in as names or labels. Thus, the emphasis in these data is on users' interactions with objects in visual direct manipulation applications. These action-level protocols capture data frequently missed by keystroke protocols. They solve the problem of cognitive interference raised by think-aloud protocols by also being passive and unobtrusive. And they address practical problems associated with volume of data by recording an order of magnitude fewer events than keystrokes. Like keystrokes, action protocols are recorded in machine-readable form, thus eliminating the need for manual transcription. Problems of analysis and interpretation remain, but they are simplified since their atomic units – actions – are composed of sequences of keystrokes or events that have already been interpreted by the application program's own command processor.

We view a protocol transcript as analogous to a statement or discourse in natural language. From this perspective, the symbols that represent individual user actions can be thought of as words, and sequences of actions as cognitive phrases, sentences, or discourse, as the user works with the system to achieve hierarchies of goals. To describe this behavior, we have developed formal analytic models expressed as grammars. We use these grammars to parse the protocol stream and to produce parse trees that reveal patterns in the user's cognitive interaction with the system. These structures represent behaviors, tactics, and strategies; the parse trees permit us to make comparisons among sessions for different users or for the same user under different conditions.

Our grammar is concerned with the task of expository writing, including planning, writing, and revising, but with emphasis on the more structural, as opposed to linguistic, aspects of the process. The grammar assumes an underlying architecture for the kinds of thinking required of this and other similar tasks concerned with developing and expressing abstract structures of ideas. The key assumption is that human beings engage a succession of different cognitive modes in order to carry out a particular task – such as writing a document or building a computer system – in accord with tactics and strategies that they have learned or developed. A mode is engaged in order to achieve a particular goal that is usually manifest as the creation of some information artifact or product, such as an arrangement of notes, a plan for a document, or a paragraph of text. To produce this product in a particular mode, people use particular sets of cognitive processes and, by implication, do not use other processes. Behavior within a mode is further characterized by sets of constraints or rules specific for a given mode. Thus, a mode can be summarized as a way of thinking engaged in order to achieve a goal, resulting in the production of a particular kind of conceptual product, as a result of using a set of cognitive processes, in accord with a set of constraints and rules. Thus, for writing, people often engage a kind of exploratory thinking early in the process that is quite different from organizational thinking in which they develop a plan to guide actual writing. These modes of thinking are different from several kinds of revision, *i.e.* structural, coherence, and linguistic editing.

Our grammar defines a particular model of writing based on this modal architecture. It views a session as being composed of a series of modes in which cognitive processes are used to produce instances of particular products or changes to products. These conceptual products are represented and developed in the computer system by the user's carrying out various operations consisting of several specific actions.

A schematic representation of this model is shown in Figure 2, where we see the four actions required to produce a node labelled n. These actions constitute an operation: create_node. In conjunction with several other operations, an existing node cluster (the product) is being expanded through the process of focused recall. All of this is taking place within an exploratory mode.

Observe in Figure 2 the five levels of protocols recognized by the cognitive grammar. At the highest level is the cognitive mode, followed by cognitive process and cognitive product. At the fourth level are the operations that create the products and themselves comprise the actions at the lowest level.

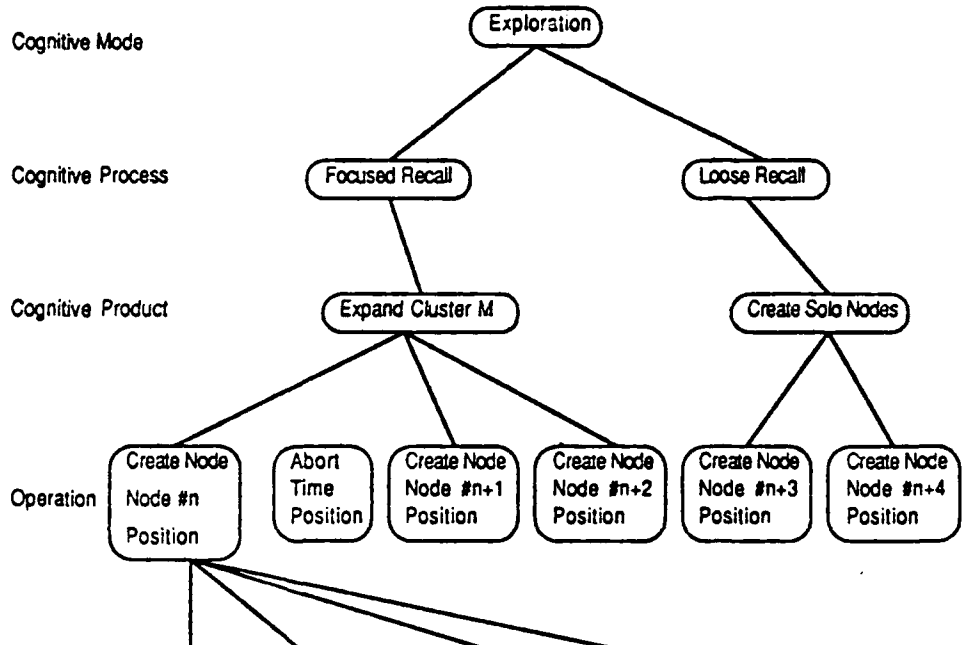

Figure 2 Example of Parse Tree

TOOLS

The events at the different levels in our model reflect various cognitive behaviors. To help researchers analyze and visualize these behaviors, we have created tools for recording, reproducing, parsing, and displaying them.

Tracking

In order to produce a machine-recorded protocol of users' interactions with a computer system, we embed sensors within our computer systems. They record data such as mouse movements, menu options and objects selected, and keyboard entries, along with the time (in milliseconds) of the action and relevant attributes, such as location of nodes. These data are formatted and written to external storage.

Data are recorded at the level of actions, which are usually complete commands, such as menu selections, or a sequence of several words, if text. Thus, the granularity of events is an order of magnitude larger than keystrokes. The reason for this difference in level of observations is that earlier research has typically focused on fine-grained analysis of user interactions, with the intent of predicting performance times and particular

sequences of low-level commands selected under different conditions. The intent of our research is quite different; we are attempting to understand broad strategies and patterns of behavior that extend over hours for complex, open-ended tasks, such as planning and writing. Actions appear to be the lowest level of event that can be said to represent an act of conceptual intent, whereas keystrokes or simple mouse movements are reflexive and, thus, lie below conceptual intent.

The transcript records, for each event, its begin-time, an empty field for the duration, the event name, and a list of attributes. The duration of the event is calculated later after the protocol has been parsed. We plan in a later version to capture and record the end-time of each action. The list of attributes is different for different actions; it consists of zero or more attributes, which may include the object of the particular action, the logical relationship of the object of the action to other objects, or information about the position of the object on the screen.

Replay

So that we may observe the behaviors of users working with our systems, we have incorporated into them facilities that take as input an action-level protocol transcript and recreate the user's actions on the display. We call this facility *replay*.

During replay, the screen appears exactly as it appeared during the user's session; however, none of the program's controls is active. Events are replicated from the beginning of the session, so the observer can watch as the user's strategy unfolds. Figure 3 shows a replay screen at one moment in time.

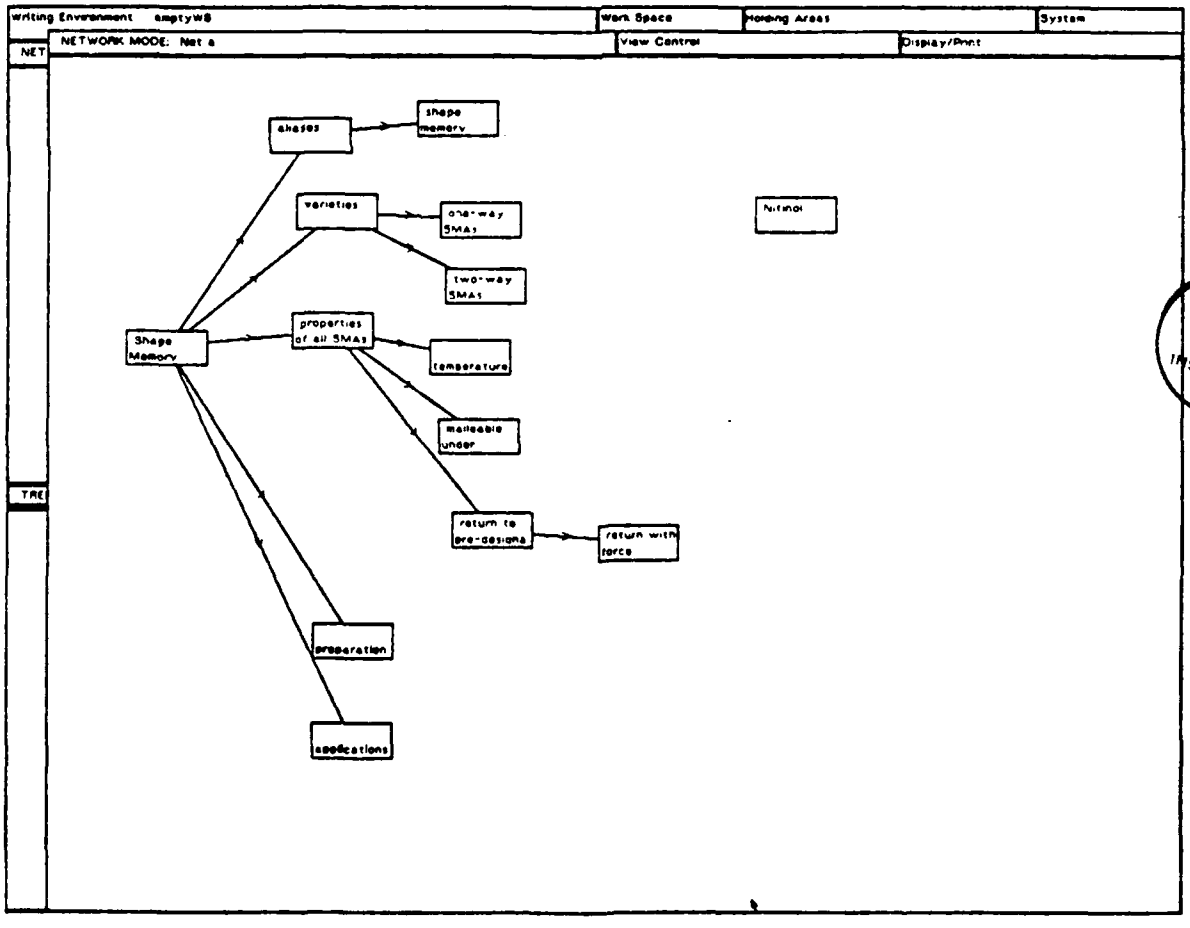

Figure 3 Replay Screen

Replay can be interrupted by the observer at any time and resumed later. The observer can also control the replay speed in several ways. The interval between events can be set so that it duplicates the duration of the original session. It can also be set proportional to the original, such as to one-tenth of the duration. These options permit the observer to gain a sense of the user's "rhythm" of work but in significantly less time than the original session. Another option lets the observer specify a constant time interval between events; this interval can range from 0, in which case the session replays as fast as the system can execute the events, to any specified number of seconds. When set to a step interval of 0, a two-hour session will replay in approximately 8–10 minutes. Finally, the observer may interrupt the replay and then manually step through the events, one at a time or a specified number of events at a time. In stepping replay, the specified number of events is shown, then a pause until the observer clicks the mouse button, then the next set of events, *etc.* Thus, replay allows the researcher to view a session in time proportional to the original, in uniform time, and in manually controlled steps.

The time compression allows one to get an over-all view of a user's strategy over a session. The visual representation allows one to see the layout of the user's nodes, the order in which ideas were generated, and the transformation of structures as work progresses. We are also able to observe the order in which different system modes were engaged and in which modes different ideas and structures were developed. At present, replay does not reproduce work carried out within the text editor; as a result, time spent in either of the editing modes is indicated by a cursor blinking for the appropriate duration, but no actual events are shown. We plan to extend our tracking and replay capabilities to show rhythms and actions for actual writing as well as for planning and structural revision.

Parsing

Our first grammar and parser were based on the formalism of production rules supplemented by functions that recognize different kinds of graph-based objects, such as disconnected sets of nodes, linked graphs, trees, *etc.* That system was implemented using the OPS83 expert system shell.

The parser is actually a group of several different parsers that operate in a bottom-up manner in which the output of one level (as shown in Figure 2) serves as the input to the next higher level. The action-level protocol transcript, produced and formatted by the tracker, is parsed to produce the operation-level protocol. The main parsing program receives these data as input and produces, in parallel, the top three levels of the parse as output: the product-level, process-level, and mode-level protocol transcripts. Thus, the parser produces horizontal slices of the parse tree that are then assembled to produce the session parse tree.

Displays

Data are available either as raw protocols or as parse trees produced by the cognitive grammar. Both forms may also be filtered to produce numerical counts, distributions across time of particular events or types of events, and other similar data. These data may, in turn, be further analyzed using standard statistical procedures.

Although data may be analyzed by numerous automated tools, they must eventually be studied and interpreted by human beings who, ultimately, decide what they mean. To assist researchers with this essential task, we are developing an open-ended collection of visualization tools. We differentiate between static tools and animated tools. The first produce a fixed image of one or more protocols, or associated data, from a specific analytic point of view. The second is set of dynamic images, usually displayed on multiple workstation screens, coordinated by a replay of a session; as the session unfolds in time, various visual displays showing the parsed or analyzed protocol update their data accordingly.

Static Displays. The Events-Time Distribution Tool, illustrated in Figure 4, provides static representations of the frequency with which user events, grouped by analyst-determined categories, are distributed over a session. The display includes three panels. In the upper left panel is a tree (represented in outline form) that defines a taxonomy of events. The frequency distributions for the events are displayed in the upper right panel of the screen. The frequencies for each particular kind of event are represented by a series of tick marks (short vertical lines). If an event continued over a period of time, a horizontal line from the center of the tick mark (the tail) is drawn to indicate the duration of the event. The panel is divided into horizontal bands that correspond to the categories defined by the tree appearing in the left panel. Tick marks appear

for each type of event and for the total for all events within a category. Changing the form of the tree in the left panel and then moving the cursor into the right panel reorganizes the right panel into new bands of events that represent the new categories of event types. The bottom panel shows a histogram that can display either the total number of events within each category or the total duration of events within each category.

In Figure 4, the events correspond to the product level of the grammar. They have been organized into three categories: explore, organize (tree), and write. In the display, the user has created exploratory products first; then he constructs the top of his tree, writes blocks of text, then goes back and fills out the bottom of the tree, and, finally, finishes his writing. In the Events-Time display of Figure 5, on the other hand, the user constantly moves between exploratory and writing operations – a markedly different strategy that is readily apparent in the displays.

To display the behaviors of several subjects so that they may be compared easily with one another, we have developed an abbreviated version of the Events-Time Distribution tool. This version organizes events into only two categories: planning – composed of exploration and organizing events – and writing – composed of writing and revision events. Figure 6 shows an example screen in which abbreviated distributions are shown for four subjects. The particular classification of events is under the control of the researcher and can be changed to fit the research perspective.

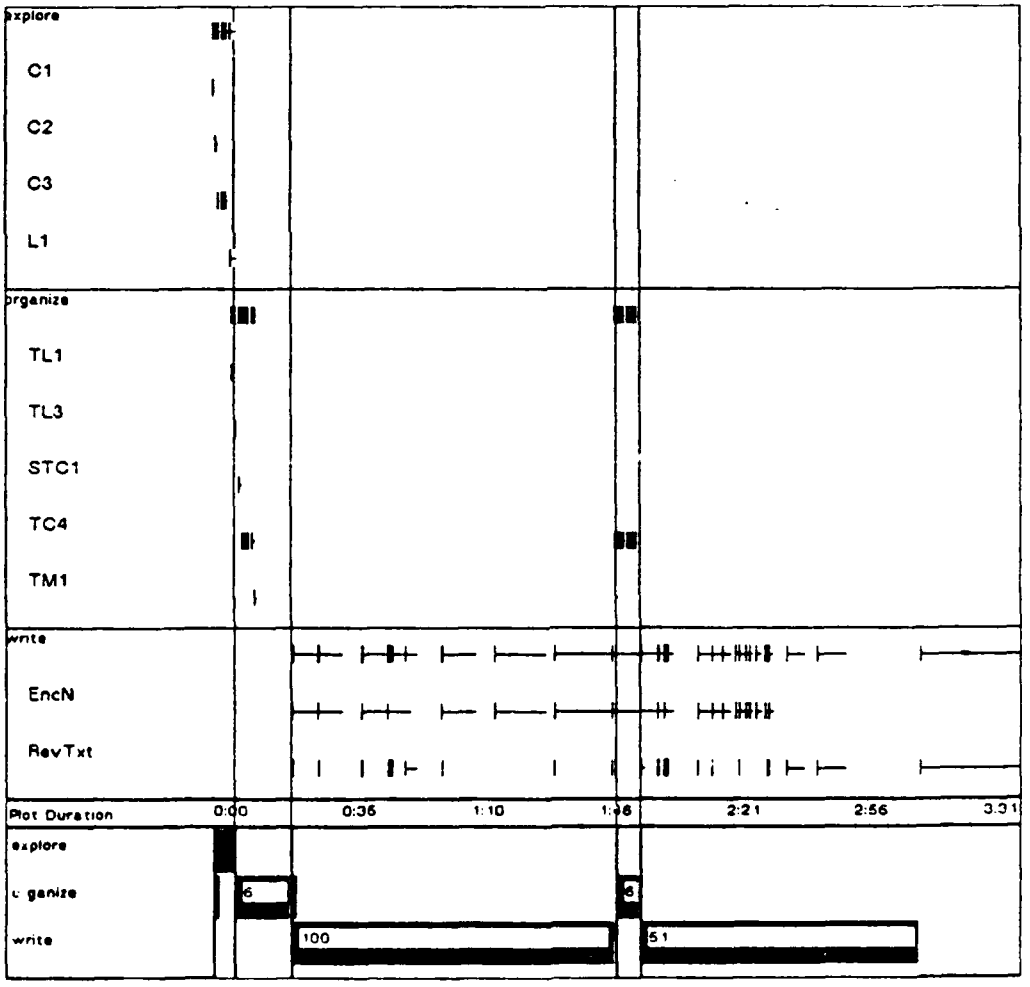

Figure 4 Events-Time Distribution: One Strategy

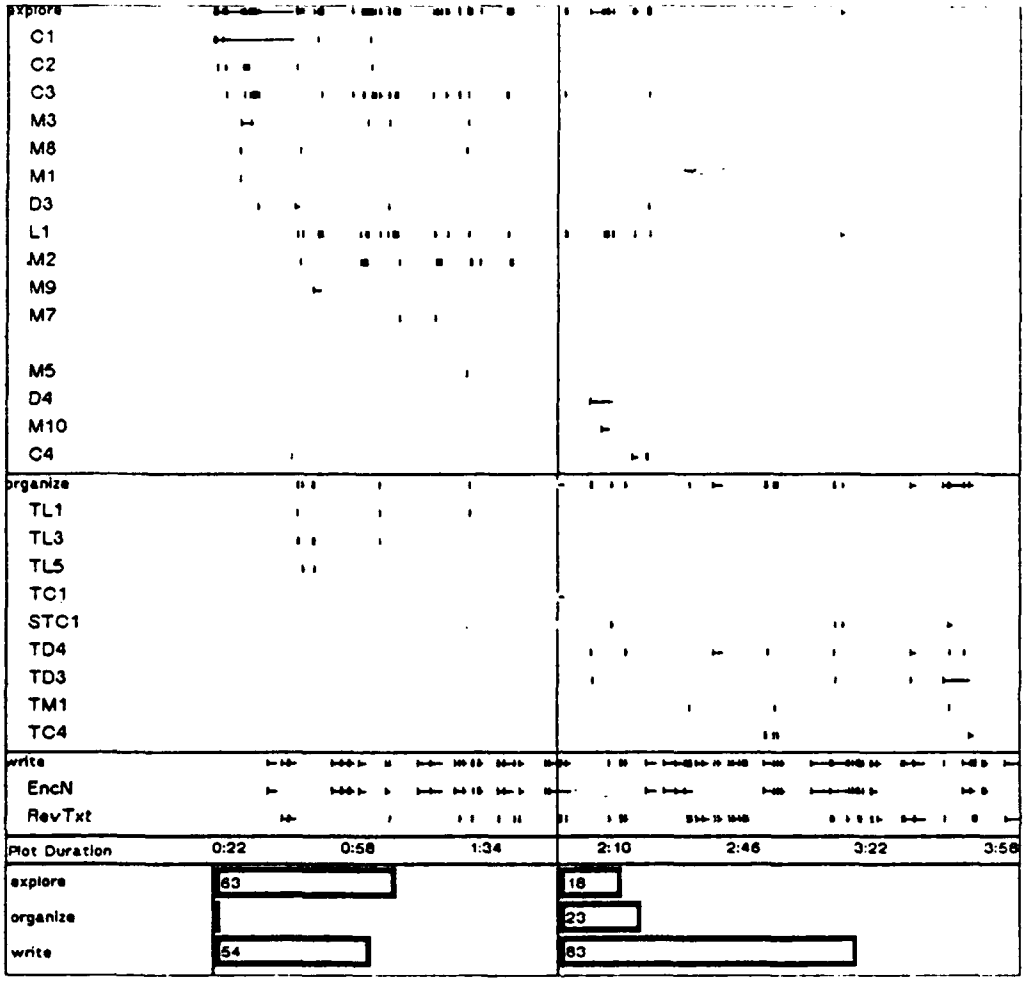

Figure 5 Events-Time Distribution: Another Strategy

Animated Displays. The data we are concerned with are inherently temporal: events take place one after the other in time. The user strategies that we study can be described as patterns in behaviors distributed in time. Thus, developing visualization tools in which time functions as the independent variable or reference dimension was a natural design choice.

The animated displays are a collection of time-oriented display tools from which the researcher may select specific displays and arrange them on multiple workstation screens to suit particular needs. We typically work with a configuration of three workstations. One workstation must be set up to run a replay of the session being viewed. It provides a central clock that synchronizes the other displays. The displays on the other workstations can be arranged to suit the researcher's preferences. They include various data displays plus control and configuration windows.

One screen in the configuration is the replay display. It is identical to the replay tool already described. Here, it serves as the central clock that coordinates all of the associated displays. Thus, as the events unfold on the replay screen, the other tools shown on the other screens update their displays accordingly.

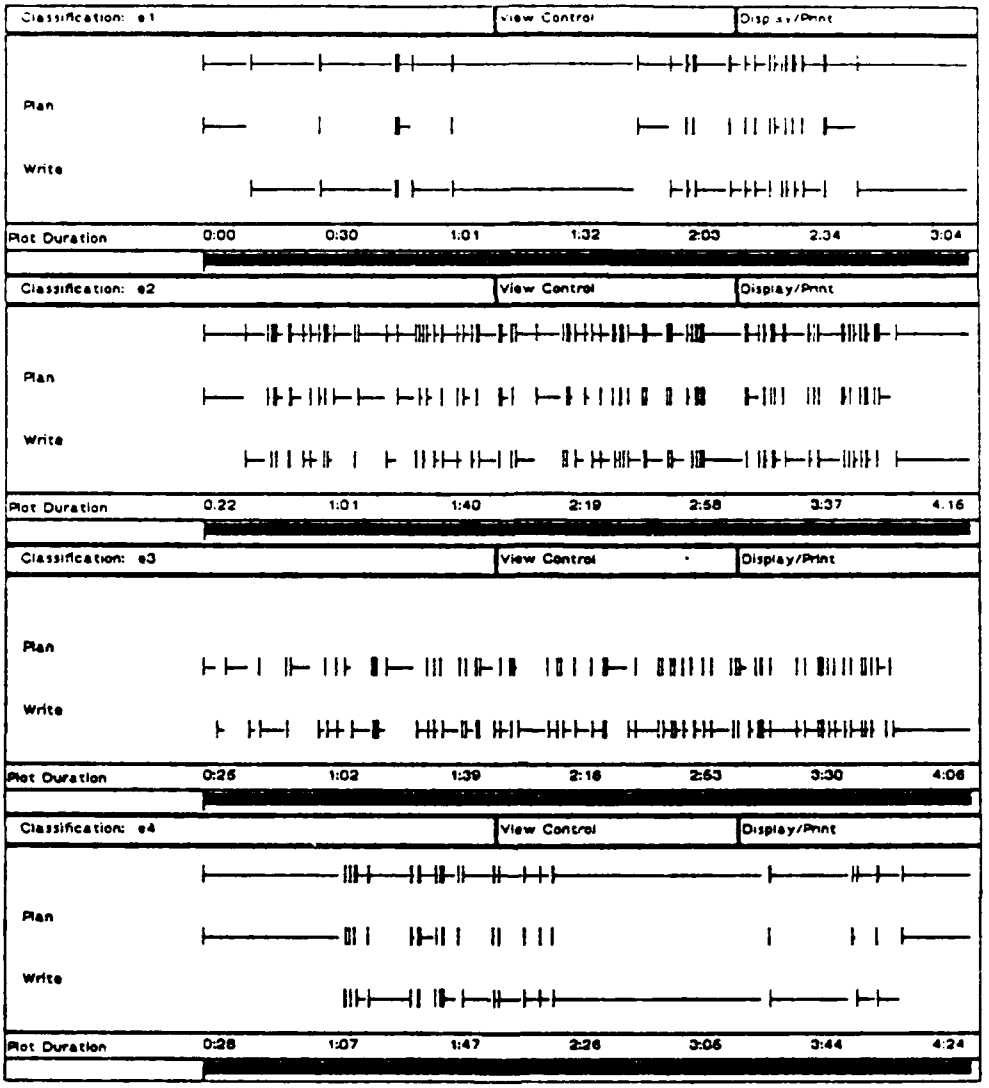

Figure 6 Abbreviated Events-Time Distributions

The animated display is controlled from a second screen, shown in Figure 7. The largest window is the configuration window, used to select the particular protocol for display and to associate specific data with the various display tools. Above that is the control window used to manage the replay tool that drives the other displays. So long as the cursor is in this window, the replay will continue; however, the researcher may stop the replay by moving the cursor to another window. Two comment windows are shown on the right side of the screen. They are text editors used to write comments or other data that are linked to a particular time or event in the protocol. Thus, the researcher may stop the animated display, by moving the cursor from the control window to one of the comment windows, and then write an observation about what was just seen in the session. When a comment window is opened, it is stamped as opening at time t_1; when the researcher no longer wants the comment shown, he or she closes the window and it is stamped time t_2. When the session is subsequently replayed, the comment will appear and be displayed from time t_1 until time t_2. The researcher can set up any number of comment windows that can be used to record different kinds of information. Thus, for example, one might have separate comment windows for a transcription of a subject's think-aloud protocol, a cued retrospective protocol stimulated by the subject's viewing a replay of his or her session, and the researchers' own observations on the session.

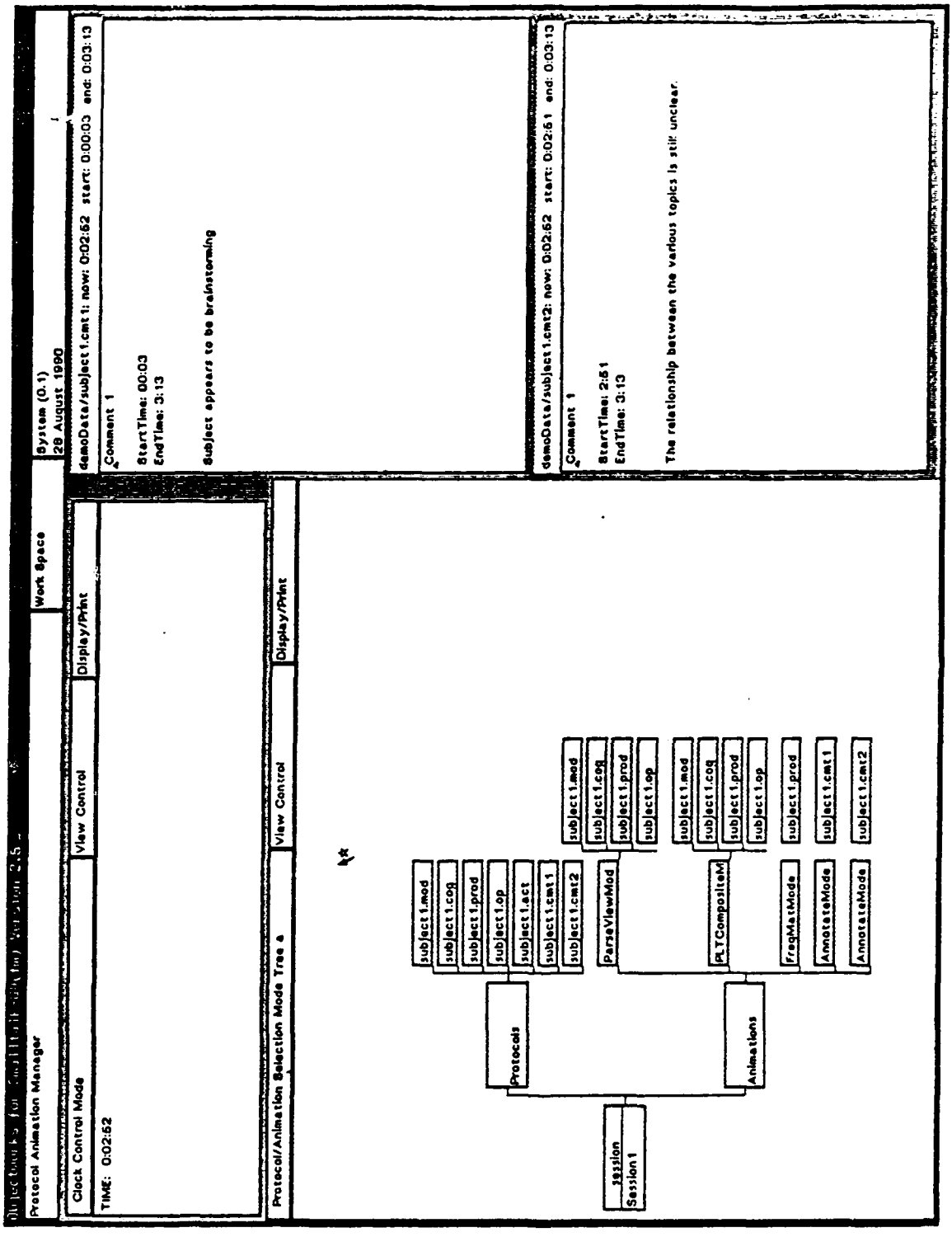

Figure 7 Animation Display: Configuration, Control, and Comment Windows

The third screen, shown in Figure 8, contains two additional animated displays. On the left is a dynamic version of the events-time distribution tool described above. Is similar to the static tool except that as the session unfolds, the vertical line shown near the left edge slowly moves across the display to the right to indicate the current time and event. A static histogram can be seen at the bottom of the display; it shows either the total time spent on each type of event or the total number of events of each type. On the right is a second window, showing a vertical slice of the parse tree produced by the cognitive grammar. It shows at the top a level for the session (not shown in Figure 2), followed by the next four levels of the parse tree: cognitive mode, cognitive process, cognitive product, and the operation associated with a particular event. The display also shows the preceding and succeeding symbols in the parse tree for each level. As the replay runs, the elements within these boxes step across the display from right to left. Using this tool, the researcher can compare the interpretation of a segment of a session, as recorded in the parse tree, with the actual events, as shown in the replay. This capability can be used for a variety of analytic purposes as well as to validate and refine our grammars.

FUTURE WORK

Group Collaboration

We will extend our tools to record, analyze, and display protocols for groups of individuals working together within a networked computer environment. This will require new techniques for integrating protocols from multiple sources, a facility for replaying group sessions, grammars that can parse both individual and group protocols, and additional methods for analyzing and displaying composite results.

We have begun to interview individuals who have worked in group collaboration projects. We have also begun observing groups, first, working "in the natural" without our computer system. We plan soon to observe artificially constituted groups carrying out assigned tasks, and, later, groups using our system to do actual work. Our experimental studies of groups will be designed to understand social, cognitive, and technological aspects of collaboration.

ATN Grammar

We are nearing completion of a second grammar expressed as an augmented transition network (ATN), and we are currently working on an associated parser. To assist with the refinement of this particular ATN model and with development of future ATN grammars, we are also building a visual, direct-manipulation grammar development tool.

Data Management

This will eventually be the central component in a comprehensive computer environment from which all protocol tools and data can be controlled. From it, we will be able to turn on and off the tracking function in our application systems; store and move protocol data; sort and select protocols from the database for analysis; develop, test, and modify grammars; parse specific protocols; apply various filter programs to the parse trees or raw protocol data to extract values that are then passed to statistical or other analytic programs; and, finally, control the various static and animated display programs.

REFERENCES

[1] J. B. Smith, S. F. Weiss, G. J. Ferguson, J. D. Bolter, M. Lansman, and D. V. Beard. WE: A Writing Environment for Professionals. *Proceedings, National Computer Conference '87*: 725–736; 1987.

[2] A. Newell and H. A. Simon. *Human Problem Solving.* Prentice-Hall, Englewood Cliffs, NJ; 1973.

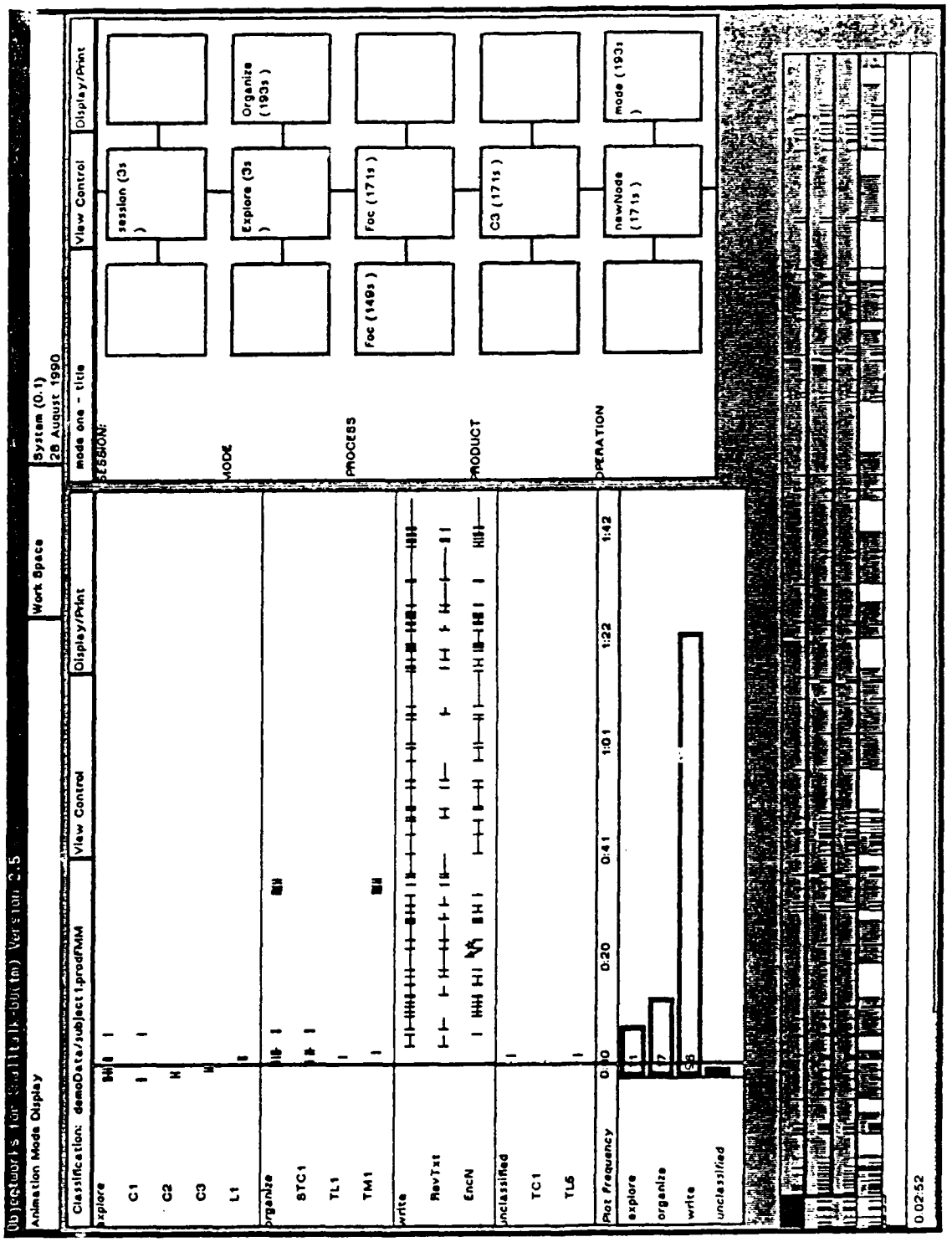

Figure 8 Animation Display: Dynamic Events-Time and Parse-Tree Windows