AD-A284 362

ENTATION PAGE. Dist: A

Form Approved
OMB No. 0704-0188

timated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, nd reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this his burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

PORT DATE: 15 JUL 1994	3. REPORT TYPE AND DATES COVERED: Final 1 MAR89 - 28 FEB 94	
4. TITLE AND SUBTITLE: SIGNET-SOFTWARE TOOLS FOR SIGNAL IDENTIFICATION USING NEURAL NETWORKS (U)		5. FUNDING NUMBERS: - C - F49620-89-0049 61102F 3005/A1
6. AUTHOR(S): Leong, H.M.F. Gevins, A.S.		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES): SAM TECHNOLOGY, INC. 101 Spear Street, Suite 203 San Francisco, CA 94105		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES): Capt. Steven Suddarth AFOSR/NM Directorate of Mathematical and Information Sciences Bldg. 410 Bolling AFB, DC 20332-6448		10. SPONSORING/MONITORING AGENCY REPORT NUMBER: AFOSR-TR- 94 0477
11. SUPPLEMENTARY NOTES: "Original contains color plates: All DTIC reproductions will be in black and white"		
12a. DISTRIBUTION/AVAILABILITY STATEMENT: APPROVED FOR PUBLIC RELEASE: DISTRIBUTION UNLIMITED A		12b. DISTRIBUTION CODE: UL

DTIC ELECTE SEP 0 8 1994 S D F

(1)

13.

We are developing a software signal processing workbench named SIGNET that simplifies exploratory analysis of multi-channel time series data. We have demonstrated, for the first time, the feasibility of building a signal- processing system around an object-oriented database (OODB). This provides a graphical means for users to create, compare, and manipulate complex data structures while maintaining system wide understanding of these structures. This understanding enables the system to provide database queries by content, data subset extraction with retention of important relationships, traceable self-documenting data, insurance that only appropriate data is fed to signal processing functions, etc. The end result is that users have a high degree of flexibility to manipulate data while data integrity and validity is protected. Over the course of the project, we completed a detailed system design, evaluated existing database technologies and chose an OODB upon which to build SIGNET. We built a prototype that implemented the essential framework of SIGNET and provided a platform with which to test the basic technical issues underlying our design. Signal review and exploratory signal analysis software was enhanced for incorporation into the SIGNET framework. We have also tested and analyzed the prototype and have found that the major drawback to our initial design was the speed of system response. The major factors causing this have been identified and speed-up solutions have been designed. We conclude that OODB technology provides a powerful and appropriate framework to model the data and processes that are used in exploratory multidimensional signal-processing applications. We are determining the commercial viability of developing the prototype into a full commercial system.

DTIC QUALITY INSPECTED 8

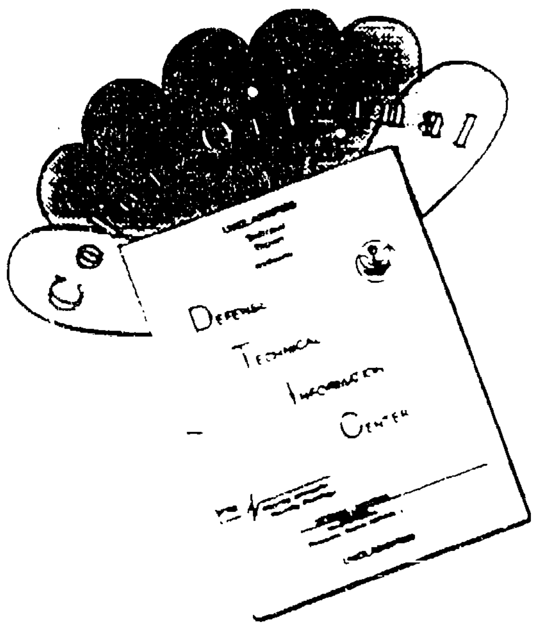
14. SUBJECT TERMS: Signal Processing Software Neural Networks			15. NUMBER OF PAGES: 36
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT: Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE: Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT: Unclassified	20. LIMITATION OF ABSTRACT: Unlimited

94 9 06 099

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF COLOR PAGES WHICH DO NOT REPRODUCE LEGIBLY ON BLACK AND WHITE MICROFICHE.

SIGNET-SOFTWARE TOOLS FOR SIGNAL IDENTIFICATION USING NEURAL NETWORKS

FINAL TECHNICAL REPORT

AFOSR Contract F49620-89-C-0049
1 MAR 89 to 30 APR 94

PREPARED FOR

Captain Steven Suddarth
Directorate of Mathematical and Information Sciences
Building 410
Bolling AFB, D.C. 20332-6448

APPROVED BY

[illegible]

Alan Gevins, President
SAM Technology

The views and conclusions in this document are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research or the U.S. Government.

Accesion For		
NTIS	CRA&I	☑
DTIC	TAB	☐
Unannounced		☐
Justification		
By		
Distribution /		
Availability Codes		
Dist	Avail and / or Special	
A-1		

DTIC QUALITY INSPECTED 3

94-29025

TABLE OF CONTENTS

SIGNET-SOFTWARE TOOLS FOR SIGNAL IDENTIFICATION USING NEURAL NETWORKS FINAL REPORT

Harrison Leong, Principal Investigator
Alan Gevins, President

BACKGROUND

The Commercial Need

Despite the variety of scientific software commercially available, there is no complete environment that integrates the full range of tools necessary for exploring multidimensional time series data. Currently, if a sophisticated analysis technique such as neural-network pattern classifier is to be used, data must be selected, extracted, converted, and imported to a specialized analysis package. Selecting and converting volumes of data between various formats required by different stages of analysis is error prone and can be prohibitively tedious and inefficient. While neural networks are potentially powerful tools for adaptive feature extraction and signal recognition, these methods must be integrated with sequences of application specific signal processing and exploratory decision making operations to effectively pinpoint the most promising signal processing approaches and transformations. Thus, all the tools used in a data analysis system need to be built around a flexible, interactive database system that facilitates data manipulation, and allows new analysis techniques to be seamlessly integrated into the software. Since most pattern-recognition research requires quick and convenient access to large quantities of raw data and results, a signal processing database must be implemented that permits researchers to track these data without needing to know complicated directory structures or file names, or whether a specific result has been calculated in advance. Such record-keeping becomes a particularly onerous task as the number of files and algorithmic variations expands. Ideally the system would understand the intrinsic relationships in the database and be capable of manipulating the data while accounting for these relationships and keeping them intact.

Providing this ideal is the central framework of a software signal-processing workbench we are developing named SIGNET. SIGNET provides specific functions for performing classification analyses on noisy, multi-channel, spatio-temporal data using neural networks. It will have a graphical environment with powerful visualization tools and a library of signal-transformation functions for exploring the characteristics of this kind of data. SIGNET will be the first system of its kind to simplify exploratory signal processing by providing powerful object-oriented database-management capabilities to organize, access, and document large signal data sets, intermediate results, and signal-processing procedures.

Object Oriented Technology

We chose to implement SIGNET using a full complement of object-oriented software technologies: the C++ language (Stroustrup, 1991); Object Interface, a user-interface toolkit; and Objectstore, an object-oriented database. This choice was based primarily on the idea that with this technology, complex software systems can be built which remain manageable and modifiable, a major problem with conventional software-development techniques. Over the long run, we would save in development costs and costs of maintaining and enhancing the software system. Many books and articles in newsletters, journals, and computer related magazines have expounded the reasons for this, especially for large, complex software systems (see for example Eckel, 1989 and Rumbaugh et al., 1991). A major difference between conventional programming and the object-oriented approach is that, in the object-oriented approach, data is bound tightly to the functions that can validly operate on the data. This minimizes invalid data operations and permits more sophisticated data abstraction, because the data elements, called objects, are defined both by the structure of their data and by specialized functionality that operates on that data. The impact of this idea on improving software development is compounded by powerful techniques to establish relationships between objects. These techniques allow programmers to literally create high-level, application-specific extensions of the language. The end result is that programming complexity can be decreased, errors can be contained, and software becomes much more expandable.

A drawback to the object-oriented approach is that using a conventional file system or database to persistently store objects is more difficult. The programmer must break up the objects and relationships between them in such a way that they can be stored in or reconstructed from a collection of inter-related data files or, in the case of a relational database, data tables. This is already a serious problem in conventional programming; a sizable portion of conventional programs is devoted to storing and retrieving data. Relationships between data must be encoded in the software rather than the data itself. Hence, maintaining the integrity of data and relationships between the data remains an insistent, time consuming task that grows exponentially with the complexity of the software application. This would be a serious roadblock for the SIGNET project. Exploratory, multi-channel signal processing involves several transformations of the data, often with input from several sources of data. The result is a complex web of inter-related networks of elemental processing functions used to generate these data. Clearly, the way in which these processing functions are connected is at least as important to a signal-processing scheme as the functions themselves. It is then essential to store these relationships. Graphical user interfaces also pressure the need to store relationships between data structures. Saving the state of a user's workspace involves keeping track of several views of the data displayed in nested graphical windows which may be sensitive to the context in which they were brought into view.

Prior to this project, we had developed a proprietary data format and accompanying data access software library which provided self documenting data, insuring data integrity and uniformity in data access by our application software. This was an important step. However, the system has no provisions to store and control interdependencies between data sets. Instead, data relationships must be encoded in the application software, in the names of data files, and in the structure of UNIX directory hierarchies. This results in an overwhelming need for good organizational skills on the part of the user; with complex analyses producing many inter-related data files, it is easy to lose track of the location of and interdependencies between sets of data. During the course of this project, it came to our attention that the technology we needed to resolve this problem could be bought in the form of a commercial object-oriented database. It appeared that this would be the most cost effective and efficient means of achieving the goals of the project.

Object-oriented databases have been developed to address the need for a seamless way to store relationships between data entities as well as the data itself. Hence, software that must access and manipulate information consisting of the relationship between two or more data elements can do so at the level of the information whereas, with conventional tools or a relational database, the software must access the data at a level underneath the target information and piece together the information desired. For example, the primary computing endeavor in which OODB technology has been successfully applied is in computer aided design applications where the inter-relationship between parts of an entity, such as an automobile, are as important or more important than the properties of the component parts. The database system we chose, Objectstore from Object Design Inc., provides C++ language enhancements and software libraries with which relationships between objects can be expressed directly and storing data is virtually transparent to the programmer. With the data storage problem resolved, programmers can almost exclusively focus their efforts on developing run-time code. However, these benefits do not come without cost. The major cost to us was that we needed to develop an object-oriented model of our signal-processing approach which encapsulated both the form of our data and the functions used to manipulate and transform it. Another primary cost is speed of data access. This concern increases with the complexity of inter-relationships between data. Hence, in our evaluation of OODBs, we placed a premium on speed. (Our evaluation of object-oriented databases is discussed below.)

SUMMARY OF ACCOMPLISHMENTS

Over the course of this project, we have succeeded in demonstrating, for the first time, the feasibility of building a signal-processing system around an object-oriented database (OODB) which provides graphical means for users to create, compare, and manipulate complex data structures while maintaining system wide understanding of these structures so that, in any given context, the appropriate form of data is used or extracted from what is given.

Specifically, we completed a detailed system design; evaluated existing database technologies and chose an OODB upon which to build SIGNET; built a prototype that implemented the essential framework of SIGNET and provided a framework with which to test the basic technical issues underlying our design; analyzed the code of the prototype to determine how system response performance could be improved; and enhanced signal review and exploratory signal analysis software, preparing it for incorporation into the system framework. We conclude that OODB technology provides a powerful and appropriate framework to model the data and processes that are used in exploratory multidimensional signal-processing applications. We are determining the commercial viability of developing the prototype into a full commercial system.

Below, structural details underlying SIGNET software will be described. The user interface will be described through a demonstration of using the system to do a pattern recognition study of electric signals from the brain. We will discuss the design changes needed to improve the core operations of SIGNET. We will describe the major software tools for multi-channel exploratory signal analysis that we have prepared for integrating into the SIGNET framework. Finally, we include a report of our evaluation of candidate commercial OODBs.

OVERVIEW OF THE SIGNET FRAMEWORK

The SIGNET framework has two primary components: a database browser, and an analysis manager. The database browser enables the user to examine and edit hierarchical data structures and specify and extract specific subsets of data for further analysis. Relationships among data elements are shown by both a hierarchical spatial arrangement and color-coding. Data is placed into the database by importing data from the UNIX file system or by executing a SIGNET analysis module. Data created by a single import operation or module execution is viewed as an atomic entity that will be called a dataset. The analysis manager allows the user to construct and execute signal-processing procedures. A signal-processing procedure is constructed by graphically linking together icons representing analysis modules which embody steps of a signal-processing algorithm. The links, called pipes, represent communication channels for directing the output of one analysis module into the input of another module. In a completely specified signal processing procedure, one or more analysis modules would have graphical input ports into which icons representing datasets would be placed. Analysis modules need not be given data in an exact form; the correct form is constructed from the input data.

Implementing the framework involved designing, coding, and testing:

1. a generic data model which could be used to represent multi-channel time series, image data, and functions that operate on these data;
2. basic data-querying operations defined on the generic data model including comparing overall structure of the data, comparing values of specific data elements, and extracting subsets of data while maintaining the correct relationships between elements of the extracted data;
3. a graphical data representation that gives immediate views of the relationships within the data and facilitates constructing queries and extracting data subsets;
4. data-structures and functions for recording and displaying the processing heritage which created a dataset;
5. graphical tools with which schematics of signal-processing algorithms can be designed, fed with appropriate parameters and data, and executed;
6. the basic elements of an automated pattern classification analysis system based on neural networks; and
7. a software library that expedites data communication between the SIGNET database, programs built on this database, and software external to the system.

THE SIGNET DATA MODEL

Considering the data-processing system as a whole, user data can be viewed as one large conglomeration of textual, numeric, and binary data. The entire conglomerate of user data in the SIGNET system is known as the "global dataspace". It is necessary to partition this conglomerate to provide a user with a comprehensible view into how the data is structured and distributed. The dataspace is logically broken into "datasets". Each dataset is created by one discrete data import operation or one execution of an analysis module such as a module for computing signal features. Data within each dataset is organized as a hierarchical structure. Each dataset contains a description of it's hierarchy and the storage type(s) of values represented at each level within the hierarchy.

For example, consider the situation of a simple electrophysiological experiment involving recordings of the electroencephalogram (EEG). Several sensors are placed on a subject's scalp and these record the electric potential fields produced on the scalp by active brain cells. Recordings are made in a sequence of discrete trials where, during each trial, the subject is visually stimulated by a trial- specific pattern to which a button must be pressed when the subject detects a special pattern. This simple situation could have the following hierarchical structure (simplified for illustrative purposes): experiment, subject, trial, stimulus, response type, response time, sensor, timeseries. The corresponding data types could be these: textual (experiment name), textual (subject name), integer (trial number), bit-map (stimulus pattern), textual (press or no press), integer (milliseconds since start of trial), textual (sensor name), and floating point (electric potential values). A graphical exposition of this structure will clarify the concepts discussed here and will be presented in the next section where we will present a pictorial demonstration of the software.

This self-describing and self-documenting data model along with supporting functions for comparing and manipulating data structures allows a given analysis module to accept many possible formats of input data. When the module begins running, input data is read from a dataset and the structure of the data can be adapted to a structure that the module can operate on. This is different from existing systems where the programmer must know the exact structure of the input data while developing the software in which case the data formats that can be represented and analyzed by the system are highly constrained. This capability of the SIGNET data model simplifies applications development and the end-user's environment by alleviating the need to worry about compatibility of data formats.

The SIGNET data-modeling system also simplifies applications development by providing a programmer's interface which closely resembles the array syntax of common programming languages such as "C", FORTRAN and Pascal. It gives a simple, intuitive syntax to index into the database.

Finally, the SIGNET data-modeling system provides a database query mechanism which uses the exact same syntax as the data model itself. Relational databases, as with most other database systems, require the user to build complex query expressions which look nothing like the syntax used to build data within the system. SIGNET uses the exact same expression syntax for query expressions as it does for modeling data. Hence, users need not learn an arcane database query language to examine the database and extract subsets of data that are of interest.

SIGNET USER INTERFACE, PROTOTYPE DEMONSTRATION

The prototype we have created will be demonstrated by a series of figures which show various stages of working with the system. The context of the demonstration is a user who already has multi-channel time series data stored in several files and wishes to analyze these data to find features of the data that have distinguishable characteristics between experimental conditions (classes of signal data). The demonstration will guide the reader through the process of importing data, browsing the structure of these data, merging data into larger data sets, extracting subsets of data, querying the database, displaying time series data graphically, processing the data to extract signal features, and analyzing patterns in these features with a neural network.

We will use data from our study of the neuroelectric signals of prolonged mental work performed in 1984 (Gevins et al., 1988, 1990). For 14 hours, five Air Force test pilots performed several sessions of a battery of five well practiced tasks. In one of the tasks, the pilots were required to view a series of digits presented one at a time, remember the last three digits shown, and, in response to each stimulus presentation, produce a finely controlled isometric finger pressure proportional to the earliest of the three digits. In 20% of the trials, the most recent digit matched the earliest of the three in which case subjects were to withhold response. In our demonstration, we will use electroencephalogram signals (EEGs) recorded during these trials to distinguish between three stages of fatigue; the stages will be defined by separating the 14 hours into three sequential periods. EEGs were recorded with 27 or 51 electrodes placed on the scalp. Eight to ten runs of 150 trials each were performed interspersed throughout the 14 hours. We will use a small subset of these data.

In this demonstration, the following system capabilities will be illustrated: importing data; constructing a signal processing schematic; getting data in and out of processing modules; graphically displaying the internal structure of data; merging data sets; graphically tracing relationships among data; querying the database; extracting a subset of data; plotting waveform data; performing a classification analysis; examining intermediate results; and examining the processing heritage of a dataset.

Figure 1 (all figures appear in Appendix A) shows the opening screen to the system. There is a "Top-level Menu", which provides access to the basic tools of the system, and a "Global Data Window" which is used to display all data sets in the database.

Figure 2 shows the process of setting up a data-processing session, specifically importing data into the system. From the main menu, "New PE Palette" was selected, generating a window which contains icons for the data-processing functions available in the system, i.e., "Processing Elements" or PEs; the window appears on the lower right of the figure. Choosing "New DPS" from the main menu generates a window within which a data-processing schematic (DPS) can be designed. In this case, the schematic is trivial and consists of only the import function. In general, a schematic is built by using a mouse to select and drag copies of PE icons into the DPS window and linking up inputs and outputs. An input is represented by a small square, a "placeholder", linked to the upper, green tab on the PE icon while an output is represented by a placeholder linked to the lower, orange tab. Inputs and outputs are linked by simply using a mouse to drag one placeholder over another. The import PE has only output since it gets its input external to the system. To specify the input source, the mouse is depressed over the import PE icon and this brings up a "dialog box" in which the user can specify a file to import and launch the import operation. In general, depressing the mouse over a PE icon brings up a "dialog box" in which parameters governing the behavior of the PE can be set and the processing can be launched.

Note that importing would normally be performed once when setting up the system with an existing set of data. From then on, this data and any results derived from it would be stored in SIGNET's database, obviating the need to reference data by file name.

The software is built around a three-button mouse. In general, the left-most mouse button is used to select and move graphical entities, select menu items, or launch processing; the middle mouse button is used to extend selections; and the right-most button is used to call up menus of actions. From hereon, when referring to mouse actions, exactly which buttons are used will be implied by context.

Figure 3 shows the result of importing three files to the system and browsing the structure of one of them, "late". Data of the fatigue study was stored in several files: one for each subject, each task, and each run. Three of these runs were imported, all from subject 7. The results of the imports appear as data sets in the "Global Data Window". The data sets have been named "early", "middle", and "late" to imply the interval of the 14 hour testing period to which they correspond. The window labeled "Unconstrained Browser" was brought up by selecting the "New Browser" option from the "Top-level Menu". The mouse was used to drag the "early" data set icon from the "Global Data Window" into the Browser window. The Browser makes a copy of the data set and displays its structure in the left panel of the Browser window, the "browsing" area. The "browsing" area is for browsing, modifying, or creating data sets. The right panel is for displaying the results of queries (illustrated below), the "data set holding" area. Alternatively, a data set can be browsed directly by clicking the mouse over its icon in the "Global Data Window" and selecting the browse option from a pop-up menu. In this case, queries and other functions of the Browser are performed in the context of the root data set.

Two simple elements of our data model have been implemented and can be seen in the Browser window. They are called "block" and "attribute" data types corresponding to the blue and brown rectangles, respectively. Each value of a "block" refers to a collection of data elements whereas each value of an "attribute" refers only to itself and is associated with other data only through data types in which it is part. This pair of simple data types can be used to represent a wide variety of data structures. The figure gives an abbreviated glimpse of how signal data might be organized in the system. It can be observed that for each trial, we have imported the time at which the trial was performed, the target behavior, variables summarizing actual behavior, and the multi-channel EEG time series. Structural elements can be added or deleted by using the mouse to access a menu in the "browsing" area of the Browser window; this menu gives the user a choice of structural elements to create, e.g., block or attribute, or deletion of a selected element. This facility also allows a user to build query expressions and data extraction templates by creating data structures from scratch and specifying values for some of the structural elements (see below).

Figures 4 a and b show the process of merging data sets imported for subject 7 into a single "subj 7" data set. The figures also illustrate how data within the structures are related. The left panel of Figure 4a reveals the data values contained in the "late" data set. (For these figures, only the relevant portion of the Browser window is shown.) Clicking on the triangles in the "block" (blue) and "attribute" (brown) icons reveals or hides data values. The right panel of Figure 4a shows results after the "middle" data set was dragged from the Global data window into the "dataset holding area" of the Browser window and selecting the "merge" command. It can be observed that data from the merged data sets came from two different "sessions", 2 and 3. Data corresponding to session 3 was highlighted by clicking on its session number; the heavily underscore indicates what was clicked, hence marking the root of the highlighting. In fact, the relationship of any data value with other values can be seen by simply clicking on the value. Figure 4b further illustrates this capability. In this figure, we have merged in the "early" data set. Notice that the "early" data comes from session 2, the same session as the "middle" dataset, and that data in the run block have been appropriately grouped; specifically, the "middle" dataset corresponds to run number 4 and this run number is grouped with run number 1, the "early" dataset's run number. In the left panel, we have clicked on one of the trials as revealed by the heavy underscore. The highlighting shows that the trial belongs to session 2, run 4, and shows all behavioral and EEG data associated with it. (There is too much time series data to display on the screen so it remains hidden.) The right panel shows the consequences of selecting a "target force" value. It can be seen that the selected value belongs to trial 68, run 6, session 3 but, except for "target force", no data values are selected following the "trials" block. This is because highlighting only shows data that contains or is contained by selected data.

This sequence of figures shows that, by building data structures using the simplest elements of our underlying data model, i.e., "blocks" and "attributes", our software has captured an understanding of the relationships between data. It is this understanding that enables our software to provide many of the benefits we have proposed, e.g., database queries by content, extracting subsets of data while retaining relationships, traceable self documenting data, insurance that only appropriate data is fed to a processing function, etc.

The final step in merging is to create a new data set in the database. This is done by selecting "Emit Descriptor" in the Browser. The data set icon then appears in the "Global Data Window"; we have labeled it "subj 7" (see far right of Figure 4b).

Figures 5a and b show the process of extracting a subset from a data set. The context is that all data from an experiment has been imported into the system and merged into one large data set. Now a user wishes to analyze a small subset of it. Imagine that "subj 7" is this one large data set and the user wishes to look for fatigue effects in trials where the subject was not required to make a response, i.e., trials where the "move type" attribute has the value "nomove". To make "subj 7," the data set from which data will be extracted, we bring up a Browser in the context of "subj 7" by clicking its icon in the "Global Data Window" and selecting the "Browse" option. We now wish to set up a data structure that will specify what data we wish to extract. Since our target data sets have exactly the same structure as "subj 7", it is easiest to make a copy of its structure as a starting point. This is done by selecting "Copy W/O Data" in the Browser. This produced the structure shown in the left panel of Figure 5a. We then create values for "run" and "move type" that will select out the desired data. The left panel shows appropriate values to select out "early", "nomove" data. The software selects data that conforms to the values specified for each structural element; if a value is not specified, any value is accepted. Choosing "Extract(start)" launches the selection. The right panel shows the result of the extraction. Two trials were found for which "move type" was "nomove" and "run" was "1". Extracting "middle" and "late" "nomove" data sets is done similarly by just changing the value used for "session". Again, "Emit Descriptor" is used to place the new data sets into the database.

In extraction, the full structure of the desired data set need not be specified. For example, the user may not interested in behavioral data other than "move type". Figure 5b illustrates extracting "nomove" time series data, ignoring other behavioral data. The top panel shows the extraction specification and the bottom panel shows the result. The software understands that the time series data would be unspecified without including the enclosing structural elements, i.e., "task", "subject", "session", "run", and "trials". Two trials from each run were found to have a value of "nomove" for "move type".

Figures 6a and b illustrate the process of querying the database. By now, we have created several data sets with slightly different structure and data values. The upper left window in Figure 6a shows a query to find all data sets which have the "target force" structural element (see the "browser" area, the left panel). The query was launched by selecting "Query (expandable)" from the browser menu. Results of the query appear in the "data set holding" area to the right. By comparing the data sets retrieved with those currently in the database (shown in the "Global Data Window" at the lower right side of the figure), it can be observed that the query retrieves all data sets but those created by the extraction of Figure 5b, as expected. Note that no data sets would have been found if "Query (fixed)" were used since this query would require an exact match between the structure of a data set and the query template. Similarly, we can set up a query by value: Figure 6b shows a query to find all data sets which have data from run 1 and 6. Again, the "Global Data Window" is shown for comparison. As expected, the "middle" data set was not retrieved since it corresponds to run 4.

Figure 7 shows a prototype processing element for graphically examining data (see the window labeled "Data Processing Schematic"). The data set to be viewed is simply dragged from the "Global Data Window" and dropped onto the input port. In this case, "subj 7" was dragged and dropped. After specifying what channels of data to display, how they are to be displayed, and launching processing, the window labeled "Wave Display" pops up. Time series can be added in separate windows or overlapped. Data can also be deleted from the display. Two waveforms have been displayed, separately and overlapped (see lower left window). The top waveform is from the "late" data set while the one below it is from the "early" data set, same channel.

Figures 8a and b illustrate setting up a more involved data-processing schematic and performing neural network classification. For this demonstration, we have imported larger subsets of data to create data sets "early nm", "middle nm", and "late nm" (see right panel of Figure 8a). They consist of "nomove" trials only. Presumably, the three data sets represent progressively increasing states of fatigue. We wish to compute RMS values in the interval 100 msec to 350 msec from the beginning of each trial and train a neural network classifier to distinguish the three states of fatigue based on these values.

Figure 8a shows the data-processing schematic. The first step to construct it is to drag the "classify" PE icon onto the "DPS window" from the palette of PE's (shown in Figure 2). At this point, the user is requested to specify the number of categories to distinguish and names for each of these; the input and output ports then appear. The three input ports labeled "early", "middle", and "late" are for inputting data with which to train the neural network. The fourth input port labeled "use" is for inputting data which the neural network is to classify after it has been trained. The next step is to drag "rms" PEs onto the "DPS window" and link their outputs to the inputs of the "classifier" PE by dragging output placeholders over input placeholders. Finally, the appropriate data sets are dragged onto the "rms" PE input ports from the "Global Data Window" and the processing schematic is launched.

Figure 8b shows the results. Intermediate results from the "rms" PEs, as well as the output of the "classify" PE, appear as new data sets in the "Global Data Window" (the lower right panel, data sets on the right). The upper left panel shows the output of the "rms" PE after processing the "late nm" data set. It can be observed that the time series have been correctly summarized by a single value for each channel. The yellow highlighting shows the "rms" values derived from data recorded during the 109th trial; the three "feat_val" values correspond to three different channels. The "classify" PE extracts the substructure consisting of "trials", "feature", "feat_val", and "class" structural elements. Hence, all signal transformation software that generate data sets for input to the "classify" PE must have this substructure. The lower left panel shows the contents of the "classify" PE output. Note that we had used the "late nm" data set as input to the "use" port so that each sample should have been classified as "late". The one sample incorrectly classified as "early" is highlighted; one error out of ten is consistent with the error rate of 0.9 reported for training under the structural element "Class Performance". In actual use, an independent data set would be placed in the "use" port to evaluate the generalization capability of the trained neural network.

Figure 9 illustrates heritage capabilities. Browsing the heritage of a data set is one of the options available in a pop up menu that is accessed by clicking over a data set in the "Global Data Window". The heritage graph of Figure 9 shows the data-processing steps (thick boarders) and data sets (thin boarders) involved in obtaining the classification result. It clearly shows that the "use rms" data set is no different from the "late rms" data set. The information required to construct this graph is automatically recorded each time a PE is used. Hence, data sets are self documenting. A heritage graph can be selected and dragged into a DPS window. This capability allows users to easily set up processing schematics by simply browsing the heritage of an example of the data set they wish to create.

Conclusions

The prototype proved that our data model and data comparison and extraction functions were functionally sound. It also showed that the mechanisms for communicating data between signal-processing functions were sound. The main problem was that system response was slow, even with the small datasets we used in the demonstration. Hence, it was clear that our design needed to be analyzed to identify how system performance could be improved. Our analysis is discussed in the next section.

EVALUATION AND EXTENSION OF THE PROTOTYPE

Code Refinement for Speed

In analyzing our prototype, we found that poor system performance was primarily caused by the operations used to compare data structures and query the database. The effects were widespread since these operations are used, not only in explicit requests by users, but also by signal-processing functions accessing data and extracting the form of data they need from the database. Our prototype implementation uses a brute-force algorithm to exhaustively applying query expressions to each dataset in the system. Our new design improves the efficiency of comparing data structures and scanning the database by optimally breaking the algorithm into functional components and re-partitioning our object-oriented design to minimize unnecessary and redundant operations. Specifically, our new design allows the programmer to view each level in the hierarchical data model as a dimension of a multidimensional array. It provides programmer tools to build a map of each data element in terms of this array so that, once the map is built, storage and retrieval is virtually immediate. From these structural changes to our data model and support functions, we expect a speed-up of at least two orders of magnitude. In addition, to maximize the ability to port our code to other commercial OODBs, our implementation did not use the build-in query facilities provided by the ObjectStore database. Using them will significantly simplify our code and may provide improved system performance.

Software Ports

We have completed an X-window/Motif port of most of our digital signal-processing software. This was a necessary first step towards integrating our current signal-processing software into the SIGNET system because the software used proprietary Masscomp functions. Below, we describe the three major applications and show screen dumps from the ported versions.

Data Reviewer

Data Reviewer (Figure 10) provides means to review and annotate recorded signals. Researchers are able to label any signal segment with pre-defined labels, e.g., eye-blink artifact. This will be generalized to handle user defined labels. It is also possible to temporarily filter, decimate, or interpolate signals using a selection of filters suitable for EEG research. This will be generalized to apply user-defined filters. Other functions include display scaling, choosing any subset of channels to display (up to limitations of screen size), and jumping to any portion of a signal file. Labels in the data files can be used by other programs to trigger special processing such as artifact rejection and artifact minimization filtering or to select out subsets of signal segments for further analysis. This latter function is provided by "Data Subdivider" described below.

Data Subdivider

This application provides means to divide the data of an experiment into subsets for hypothesis testing using any of the following alone or in combination: variables which characterize the circumstances in which signals were recorded, labels placed in "Data Reviewer", and characteristics of the signals themselves such as power in a specific passband. Data Subdivider does this by allowing researchers to numerically or graphically specify values or ranges for these criteria, and then immediately view the effects of their decision on histograms showing the numbers of signal segments satisfying the criteria (Figure 11). To aid researchers in specifying appropriate criteria, t- and F-statistics are computed. The software writes out a file which contains descriptions of the constituent trials and information about the selection criteria. These output files are used to guide other programs in retrieving data from the recording that fulfills specific selection criteria. With this functionality, Data Subdivider provides the means by which researchers can insure that the hypotheses they aim to test are actually those that are tested. It does this by providing the means to eliminate sources of variance in the data that are irrelevant to the hypothesis being tested.

EXPLORE

The purpose of EXPLORE is to provide the researcher with graphical tools to review and compare processed data using a number of graphical representations. EXPLORE has the ability to make 2-d plots, color-coded topographical maps on 2-D projected spherical surfaces, and specialized plots such as spatial cross-covariance (Figure 12). Researchers are able to obtain numerical read-outs of function values on 2-D plots. Plots can be sized, placed, moved, and superimposed on other plots anywhere on the screen using a mouse, much like graphical entities in object-oriented drawing software. Researchers can flexibly and easily scale plots, label axes, and obtain print outs.

Integration with 3-D Visualization Software

One goal of SIGNET was to integrate our database and data modeling system into a numerical analysis and 3-D visualization system such as Explorer from Silicon Graphics, Inc. or AVS from AVS, Inc. We chose to study integration with Explorer from Silicon Graphics because Explorer provides the most advanced visualization functionality and the best distributed processing environment.

We have completed our investigation into integration with the SGI Explorer system and have concluded that several approaches are possible. The tightest degree of integration would be a very close coupling between Explorer, the SIGNET data model and the ObjectStore database. This would involve replacing Explorer's data access facilities with the SIGNET data model. The most flexible degree of integration would not alter Explorer's data modeling facilities. Integration between Explorer modules would be accomplished by providing import/export functions which would provide access between each Explorer module and the SIGNET data model. These import/export functions would convert data in the SIGNET database into and out of the Explorer data management system. This is the more likely route for us to take in Phase III since tightly coupling Explorer and SIGNET would require considerable support from Silicon Graphics Inc. and Object Design.

OBJECT-ORIENTED DATABASE EVALUATION

We evaluated six object-oriented databases: Objectivity from Objectivity, Inc.; Objectstore from Object Design, Inc.; Versant from Versant Object Technology; Gemstone from

Servio Corp.; Ontos from Ontologic, Inc.; and Orion from Itasca Systems, Inc. Nineteen issues were evaluated. For each issue, products were rated on a scale of 0 to 10 with 0 denoting unacceptable and 10 denoting excellent. Table 1 summarizes the results. Each issue is described briefly below. Object Design's Objectstore was the clear winner.

TABLE 1: Object Oriented Database Evaluation Summary[1]

Issue	Weighting	Object Oriented Database Product					
		Objectivity	ObjectStore	Versant	Gemstone	Ontos	Orion
PC Platform	8	4	7	1	5	9	0
Complete C++	10	5	10	8	2	7	1
Development Environment	7	10	6	8	8	8	0
Conceptual Simplicity	10	3	10	5	1	5	2
Quality of DML	10	7	9	2	5	4	6
Size	7	10	7	10	10	10	10
Schema Evolution	6	5	7	9	0	7	10
Concurrency Control	5	10	10	0	10	10	0
Versioning	10	10	10	10	0	9	10
Indexing	8	3	8	8	5	8	5
Performance	8	4	8	3	?	5	?
Database Browser	10	9	8	6	8	8	0
Schema Designer	10	8	10	6	8	7	5
Other Goodies	10	0	8	5	3	3	0
Development Price	10	6	7	3	8	7	9
Resale Price	10	?	5	1	?	9	?
Support Quality	10	3	10	8	8	8	?
Cost of Product Failure	10	8	6	8	8	7	9
Standards	10	8	10	6	3	2	0
Weighted Average Score[2]		6.2	8.3	5.7	5.3	6.9	4.3
Weighted Average Range[3]		5.8-6.4	8.3-8.3	5.7-5.7	4.8-5.8	6.9-6.9	3.6-5.2

1 All numerical entries are ratings on a scale of 0 to 10 with 0 being low.
2 Values that could not be determined, (?), were simply ignored.
3 The lower bound was computed by assuming indeterminate values were 0, upper bounds by assuming 10.

Description of Issues in Table 1

PC Platform: Are there plans for porting the product to the IBM-PC platform? Ultimately, we would like to run SIGNET on low-cost IBM-PC compatibles.

Complete C++: How well does the product support the C++ language? For example, some did not support multiple inheritance. All but Objectstore did not support parameterized types, a powerful, labor saving feature.

Development Environment: What software development environments can be used with the product? We were looking for the most flexibility in choosing environments.

Conceptual Simplicity: How simple is it to store something in the database and navigate through the database? We sought an approach that could be easily understood to minimize programmer learning curves and maximize maintainability of the software code.

Quality of Data Manipulation Language (DML): The DML is the facility for querying the database and maintaining the integrity of relationships between objects. How powerful is the DML and is it conceptually simple and easy to use?

Size: Is the maximum database size restrictive?

Schema Evolution: When the structural design of objects change, how easy is it to update old data in the database to this new structure?

Concurrency Control: There are two control options, pessimistic and optimistic. With pessimistic concurrency control, objects in use are locked out from being used by other users until checked back into the database. Optimistic control is better suited to our application since it involves less computational overhead and the risk of conflict is low.

Versioning: Does the product support different versions of an object and can the versioning history have multiple branches?

Indexing: How efficiently can the product handle standard, non-object, data? We were looking for the inclusion of efficient algorithms to deal with standard data, i.e., B-trees and hash tables.

Performance: How fast is data access?

Data Browser: How good are the facilities for examining the contents of the database?

Schema Designer: How good are the facilities for implementing the structural design of objects with output to C++?

Other Goodies: Are there other helpful tools that come with the product, e.g., third party software libraries.

Development Price: Price of the product for software developers.

Resale Price: Price of run-time licensing when SIGNET is packaged and sold.

Support Quality: The responsiveness of their technical support group.

Cost of Product Failure: How difficult would it be to convert our code to work with another company's product in the event the product fails in the market place?

Standards: How committed is the company to industry standards?

CONCLUSIONS

We have laid the groundwork for creating a multidimensional signal-processing software workbench based on an object-oriented database. We have observed that the major drawback to our initial design was system response performance. The major factors causing this have been identified and solutions have been designed. To significantly improve system response performance, a few major structural changes to the code will need to be implemented. Integrating our exploratory signal analysis applications into the SIGNET framework would best be done following these structural changes. We must seek further funding to complete these steps.

REFERENCES

1. Gevins, A.S., Bressler, S.L., Cutillo, B.A., Illes, J., Fowler-White, R.M., Miller, J., Stern, J., Jex, H. (1990) "Effects of prolonged mental work on functional brain topography.," *EEG clin. Neurophysiol.*, 76, pp. 339-350.

2. Gevins, A.S., Cutillo, B.A., Fowler-White, R.M., Illes, J. & Bressler, S.L. (1988), "Neurophysiological patterns of operational fatigue: preliminary results,". *NATO/AGARD Conference Proceedings*, 432, pp. 22-1 to 22-7.

4. Eckel, B., (1989), *Using C++*, Osborne McGraw-Hill, Berkeley, CA.

5. Rumbaugh, J., Blaha, M.., Premerlani, W., Eddy, F., and Lorensen, W., (1991), *Object-Oriented Modeling and Design*, Prentice Hall, New Jersey.

6. Stroustrup, B., (1991), *The C++ Programming Language*, 2nd ed., Addison-Wesley Publishing Co.

APPENDIX A: Figures

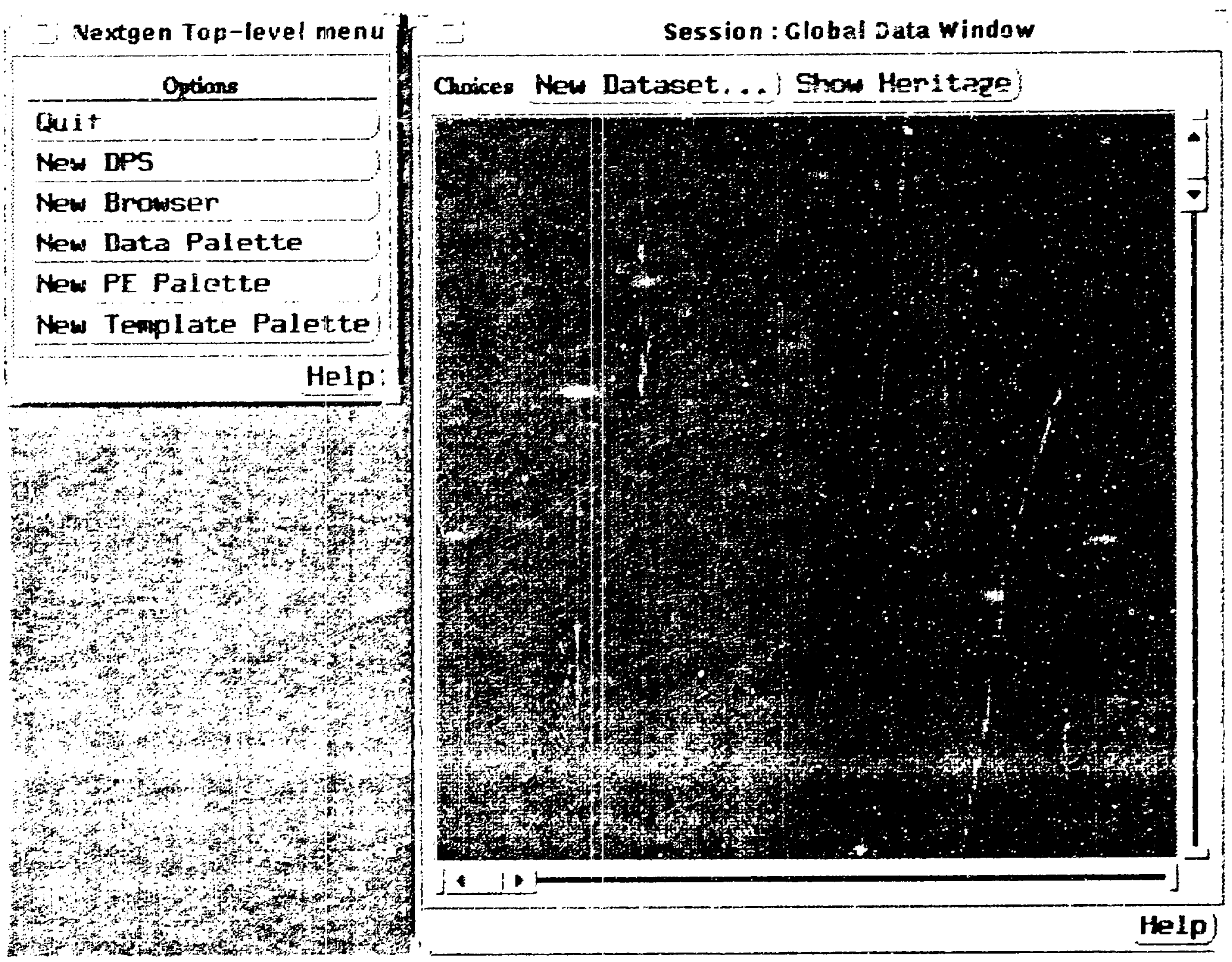


Figure 1: Opening Scene for SIGNET demonstration.
The figure shows the top level menu (left panel) and the window that will display icons for all data sets in the database (right panel). Note that, our internal name for SIGNET was the Nextgen project.

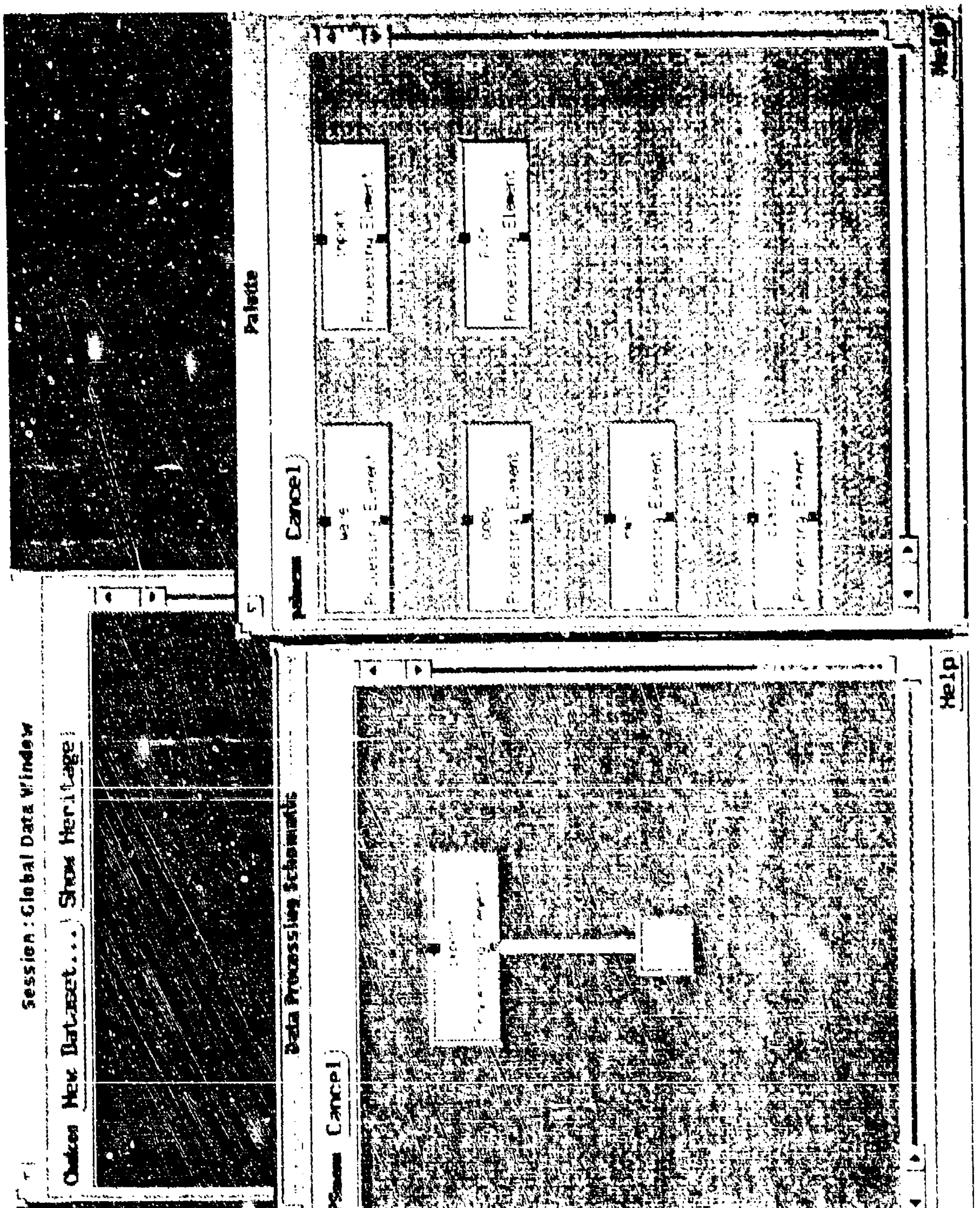


Figure 2: Using Data Processing Functions.
The figure shows a function for importing data to the system (lower central panel) and a palette containing all processing functions currently available in the system (lower right panel).

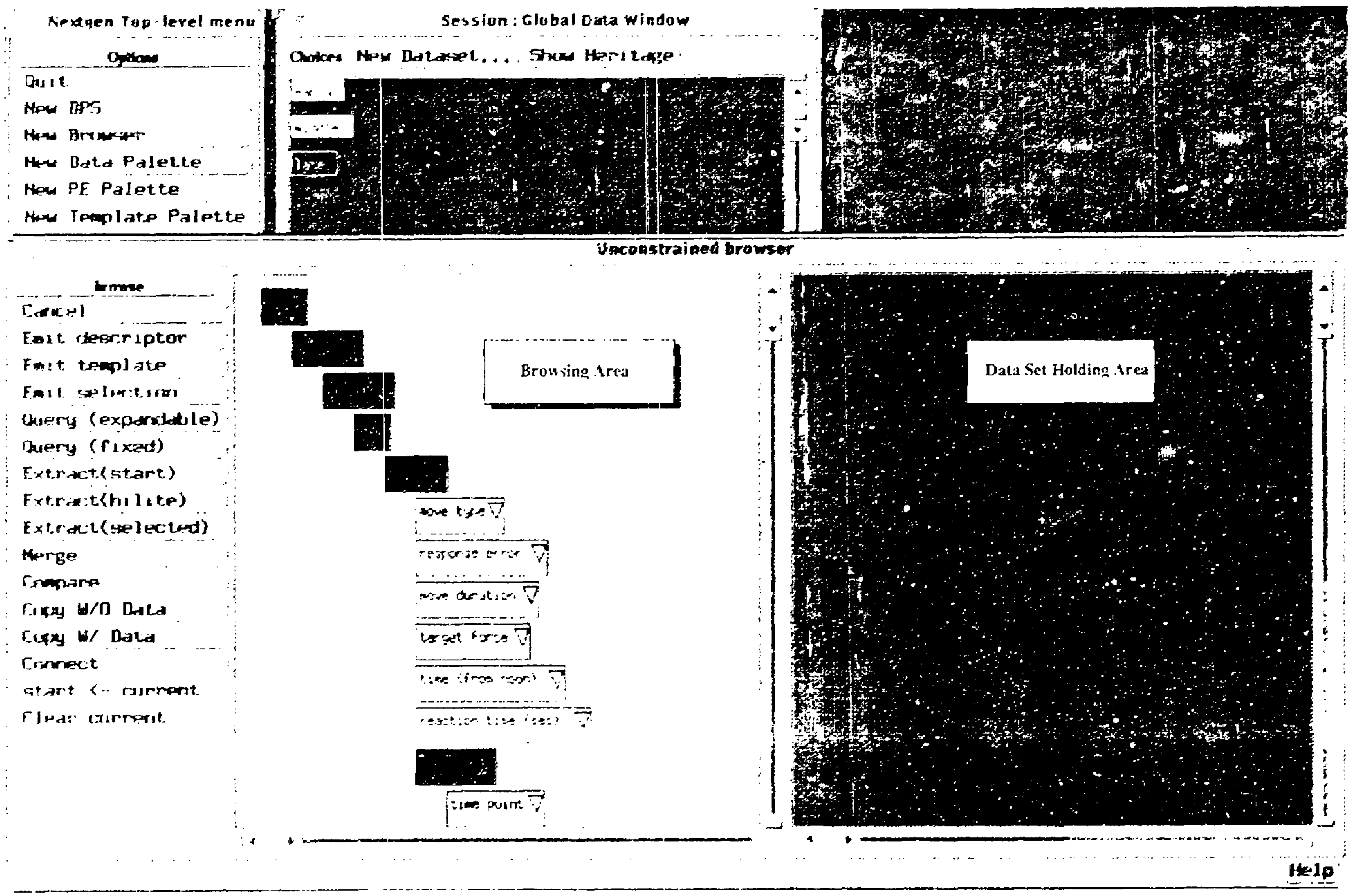


Figure 3: Example of Browsing a Data Structure.
The "late" (blackened icon) data set is shown. This structure signifies data collected during a brainwave experiment. The blue icons represent structural elements that relate collections of elements, the "block" data type, while brown icons are isolated values, the "attribute" data type.

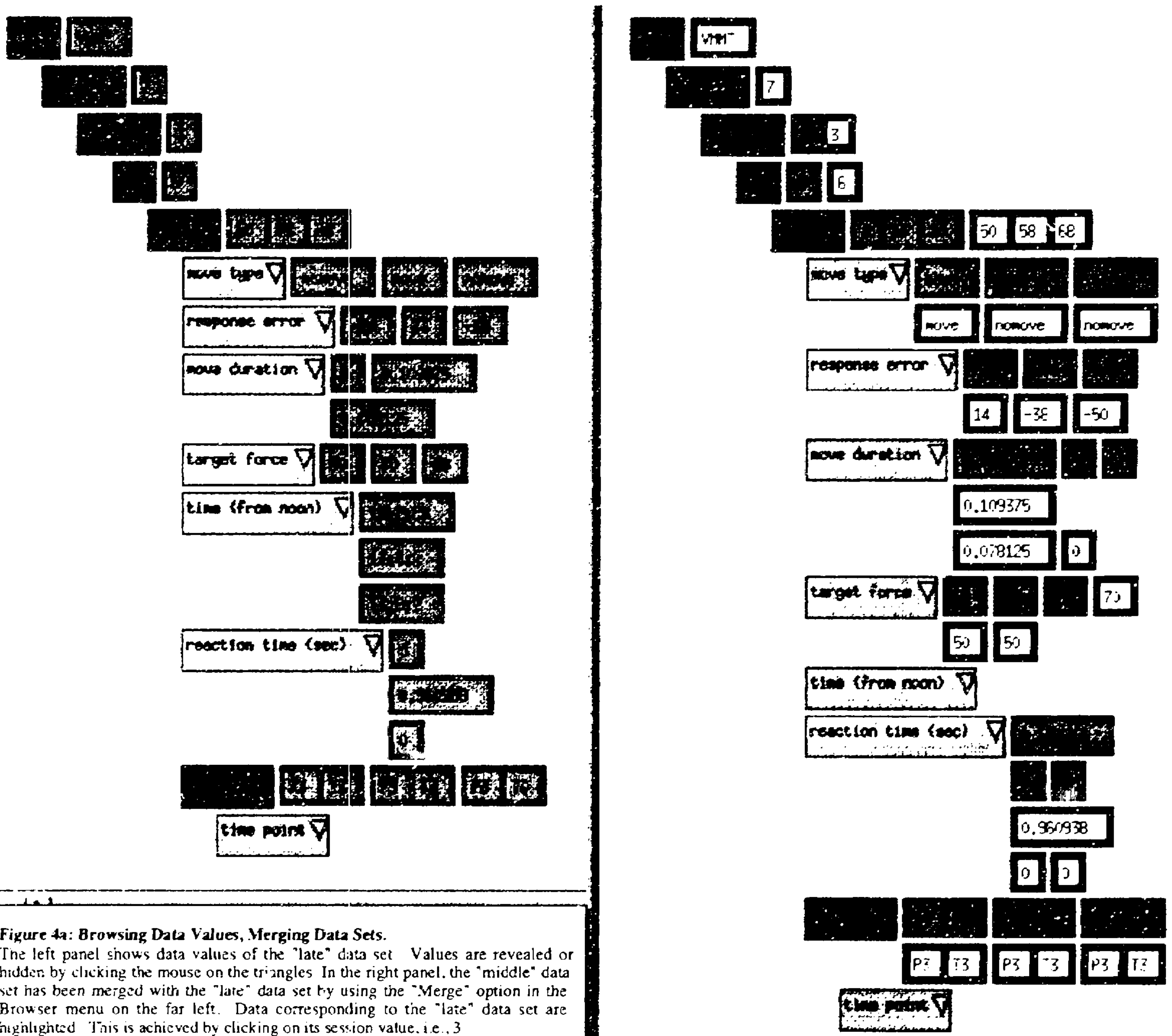


Figure 4a: Browsing Data Values, Merging Data Sets.
The left panel shows data values of the "late" data set. Values are revealed or hidden by clicking the mouse on the triangles. In the right panel, the "middle" data set has been merged with the "late" data set by using the "Merge" option in the Browser menu on the far left. Data corresponding to the "late" data set are highlighted. This is achieved by clicking on its session value, i.e., 3

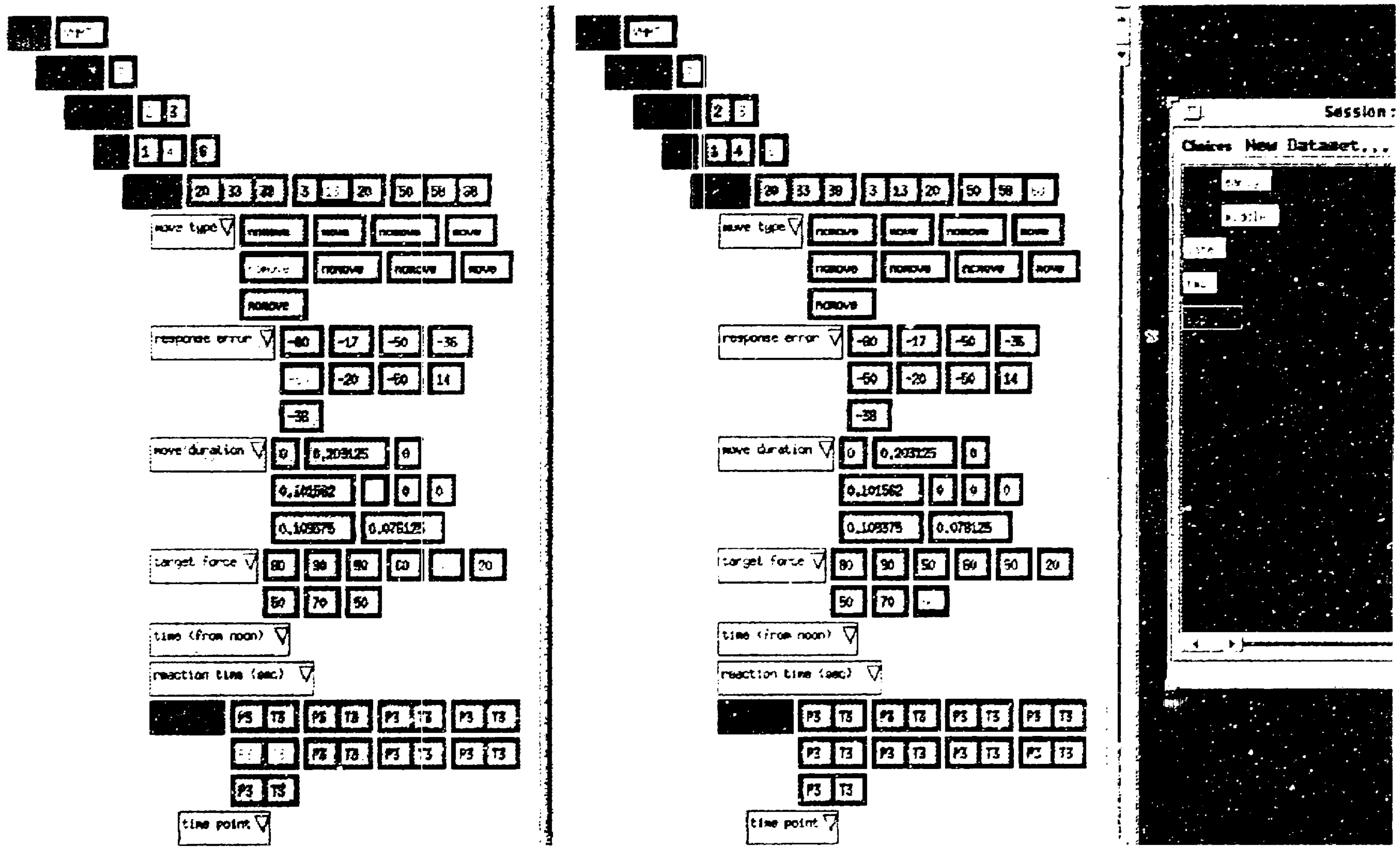


Figure 4b: Merging Data Sets, Data Value Relationships and Data Types.
The "early" data set has been merged in. The data set has been placed in the database as "subj 7" (rightmost panel) by selecting the "Emit descriptor" option from the Browser menu . In the left panel, data corresponding to trial 13 is highlighted, further illustrating the software's ability to understand the relationships between data values. The middle panel shows the consequence of clicking a "target force" value. The context in which it belongs becomes highlighted. Other values are not highlighted because it is not the responsibility of "target force" to maintain relationships between them; this is the responsibility of the "trial" structural element.

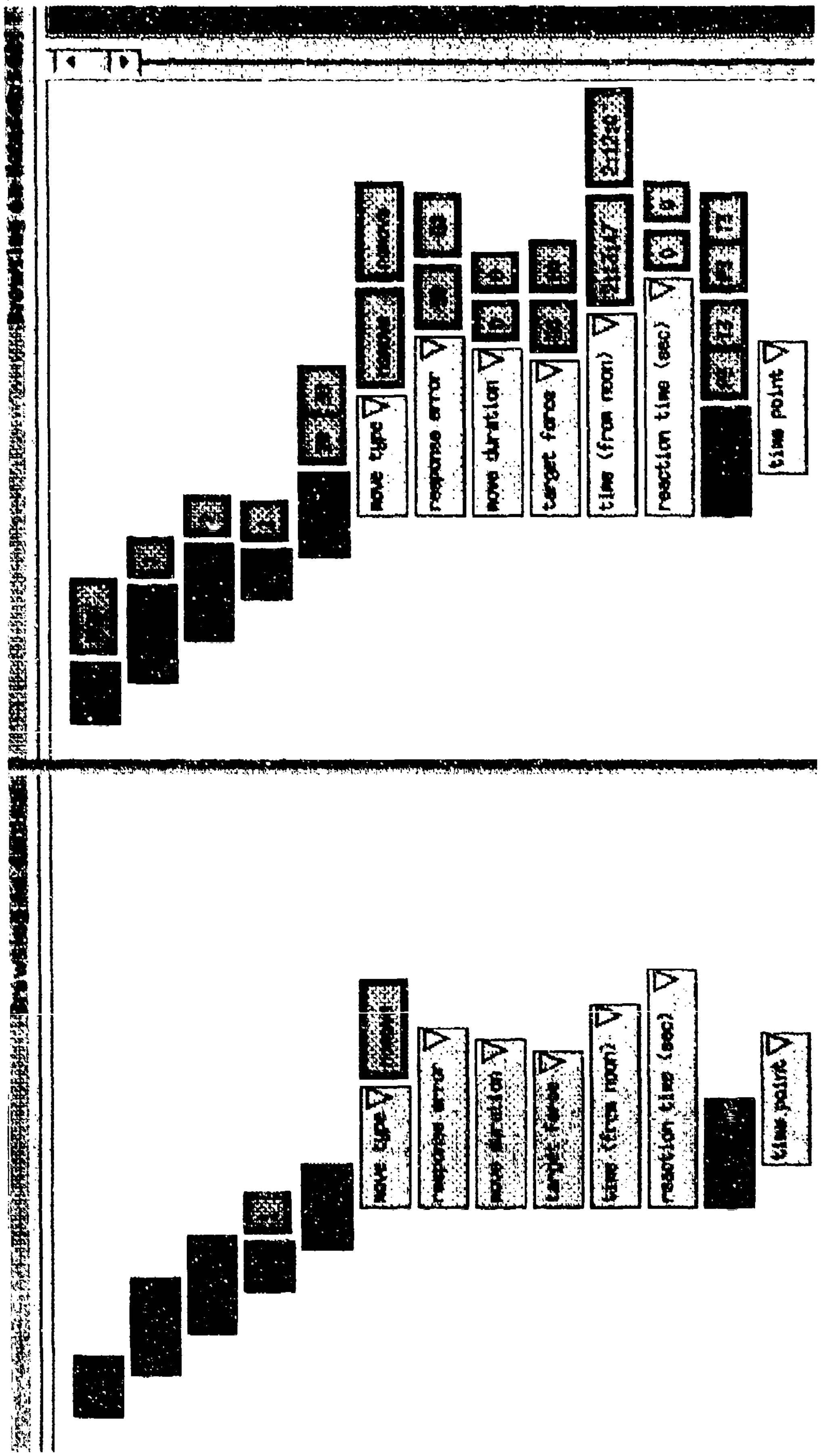


Figure 5a: Extracting a Subset of Data.
The left panel shows the data description used to extract data from run 1 for which "move type" is "nomove". Results of the extraction are shown in the right panel. The "Extract(start)" option was selected from the Browser menu.

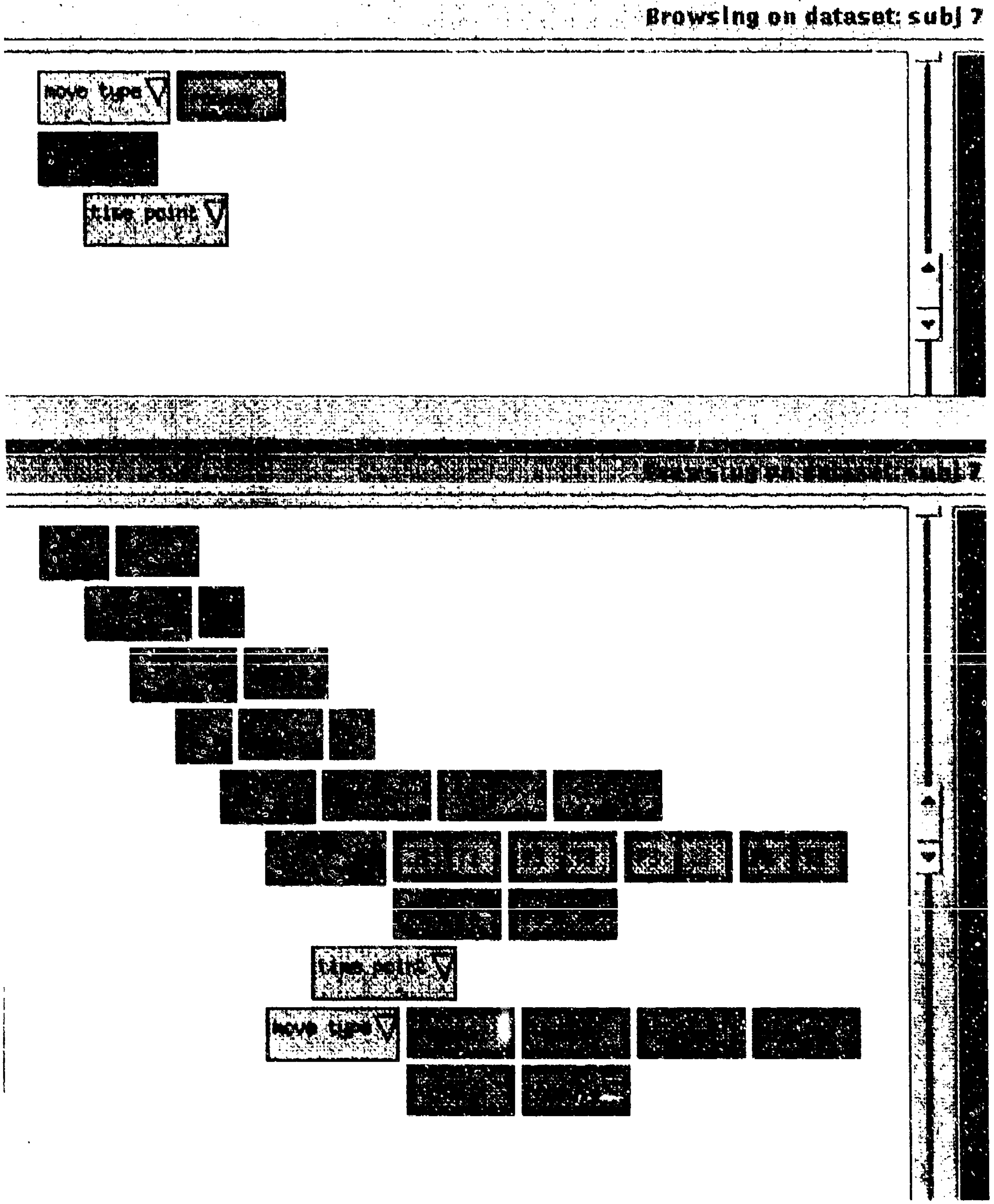


Figure 5b: Extracting a Subset of Data.
The top panel shows a minimal data description for extracting "nomove" data, including time series data but ignoring other data. The bottom panel shows the extraction result.

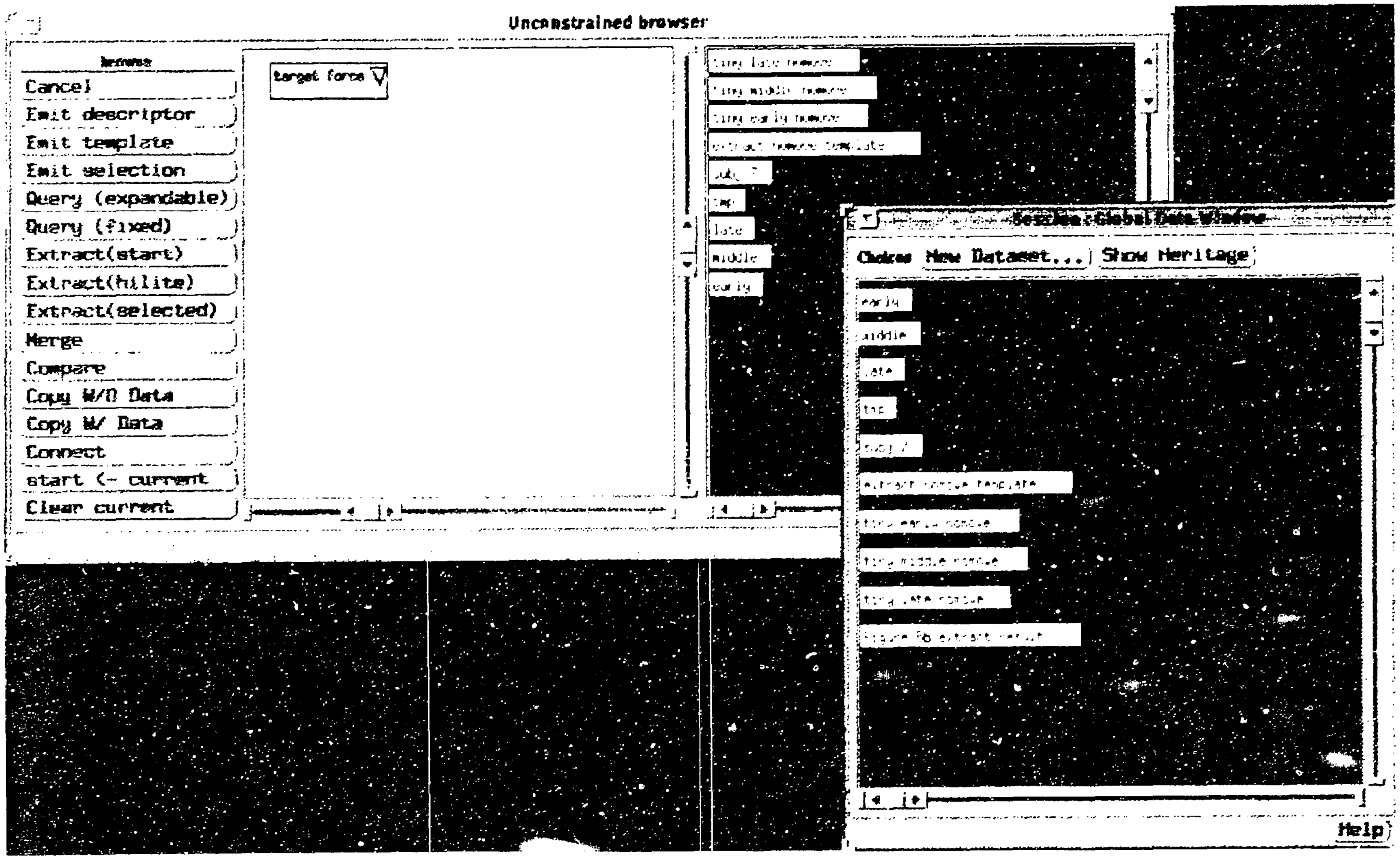


Figure 6a: Querying the Database, Structural Query.
The left side of the Browser window shows the query expression while the right side shows the query result. The "Query (expandable)" option was selected. The lower right panel shows all data sets in the database for comparison with the query result. The query found all data sets with structural element "target force".

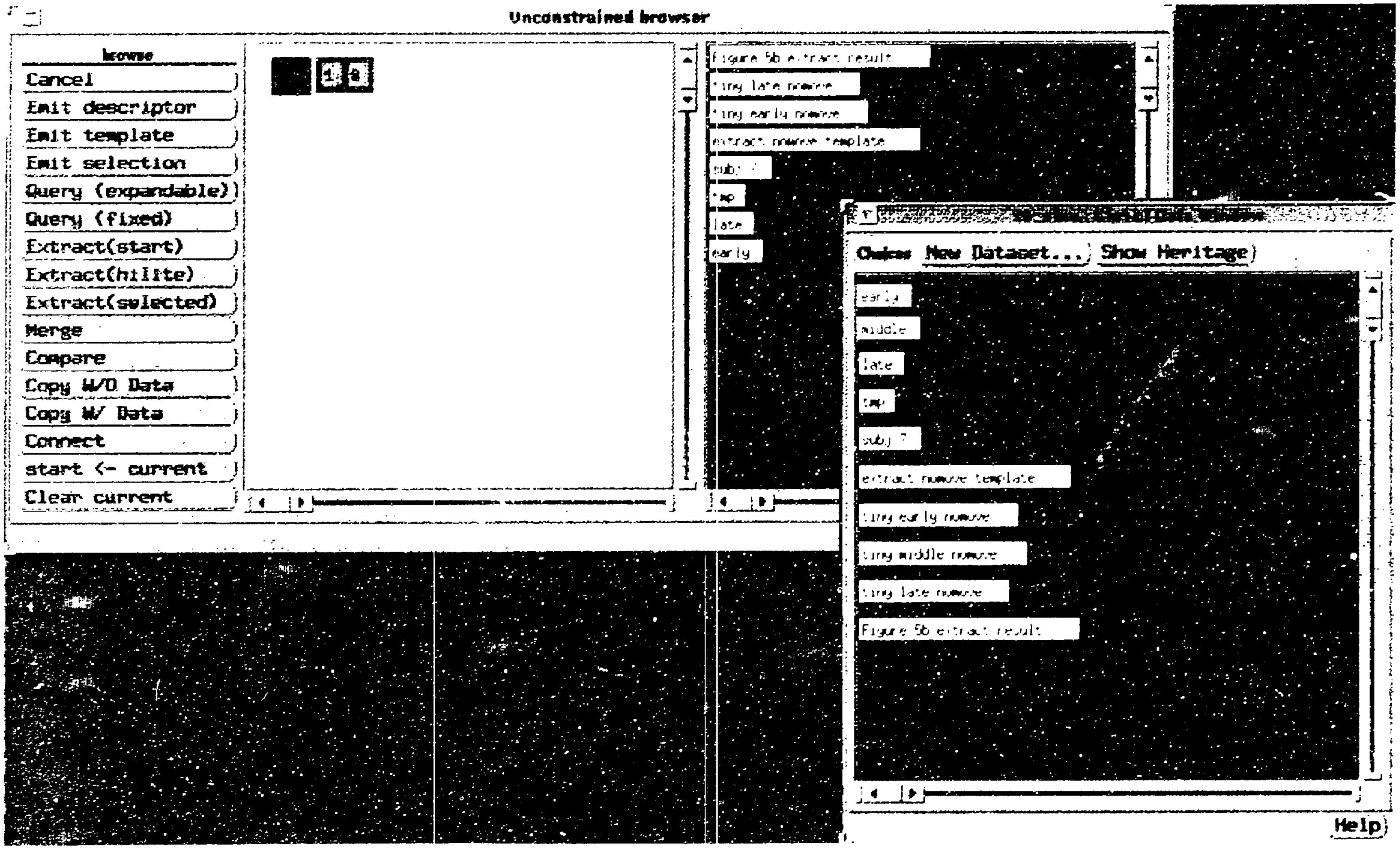


Figure 6b: Querying the Database, Query by Value.
Similar to Figure 6a except that the query expression specifies finding all data sets that have run 1 or run 6.

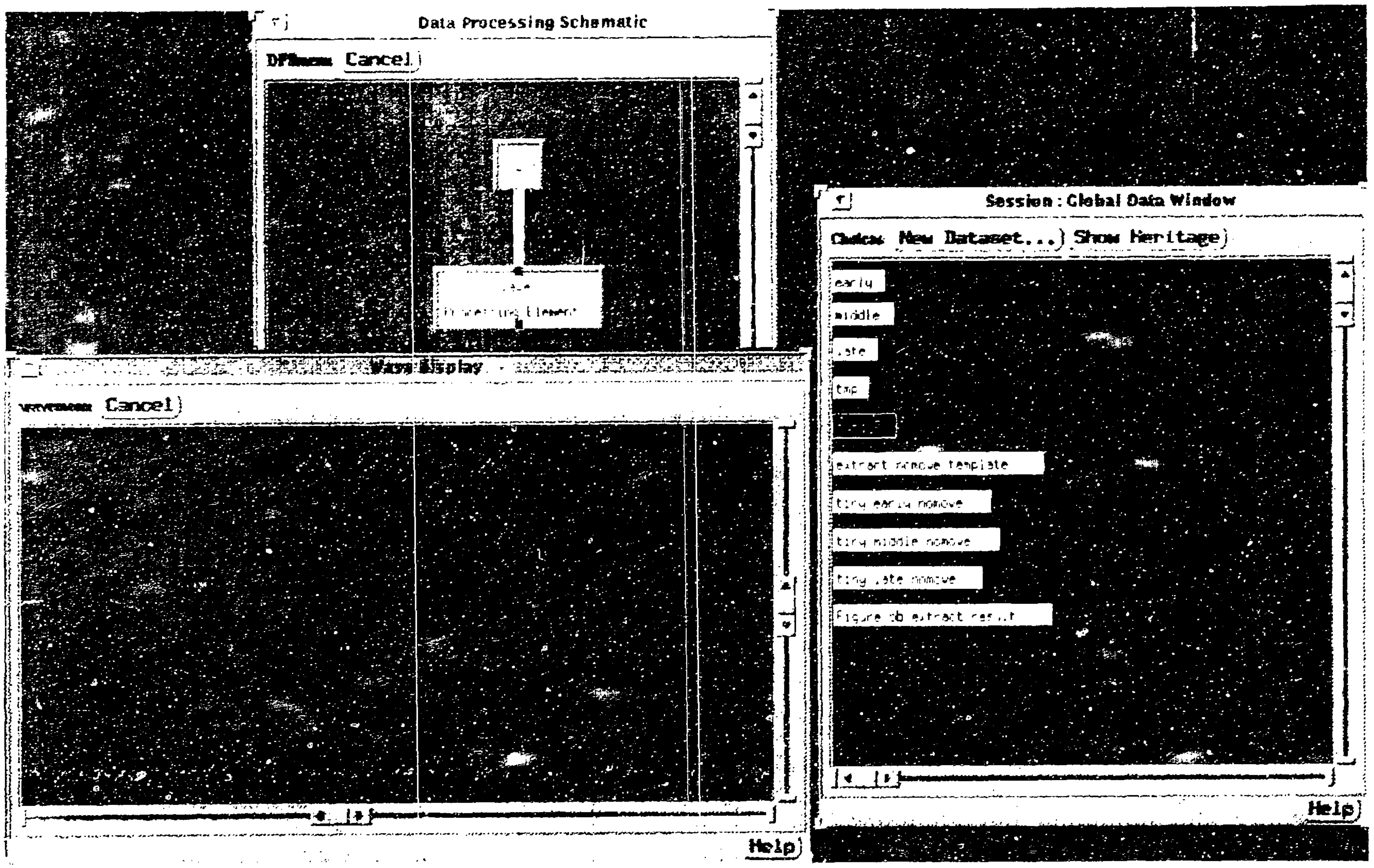


Figure 7: Graphical Time-Series Display.
The results of the processing element for graphically displaying data is shown. Two time series from the "subj 7" data set are shown separately and overlapped in the lower left panel. The top tracing is from the "late" run while the middle tracing is from the "early" run.

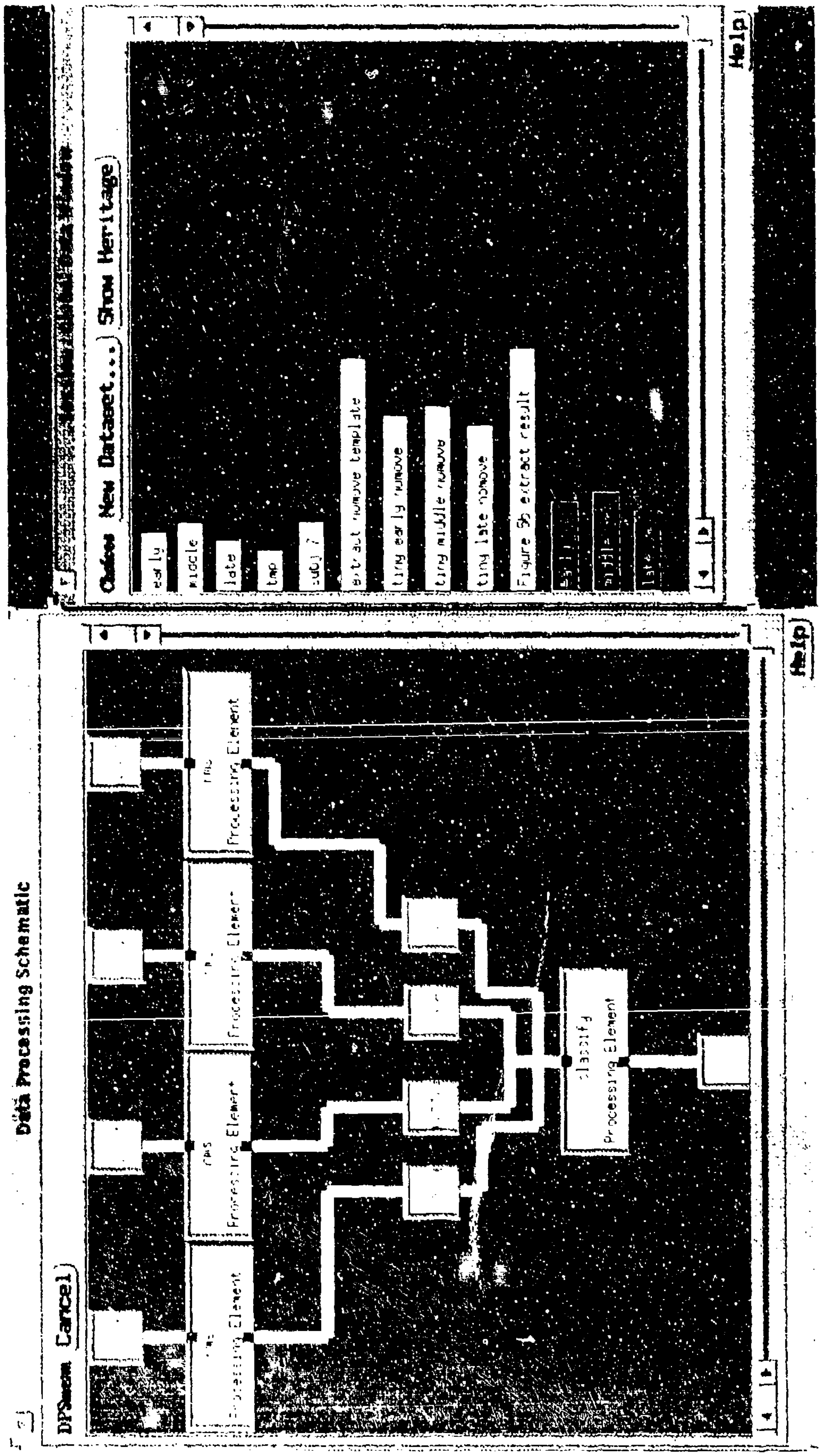


Figure 8a: Neural Network Classification Processing Schematic. The left panel shows the processing schematic set up to compute RMS values with which a neural network is trained to distinguish "early", "middle", and "late" data. The "use" node is for inputting data for a trained neural network to analyze. The three selected (blackened) data sets in the right panel were used for training.

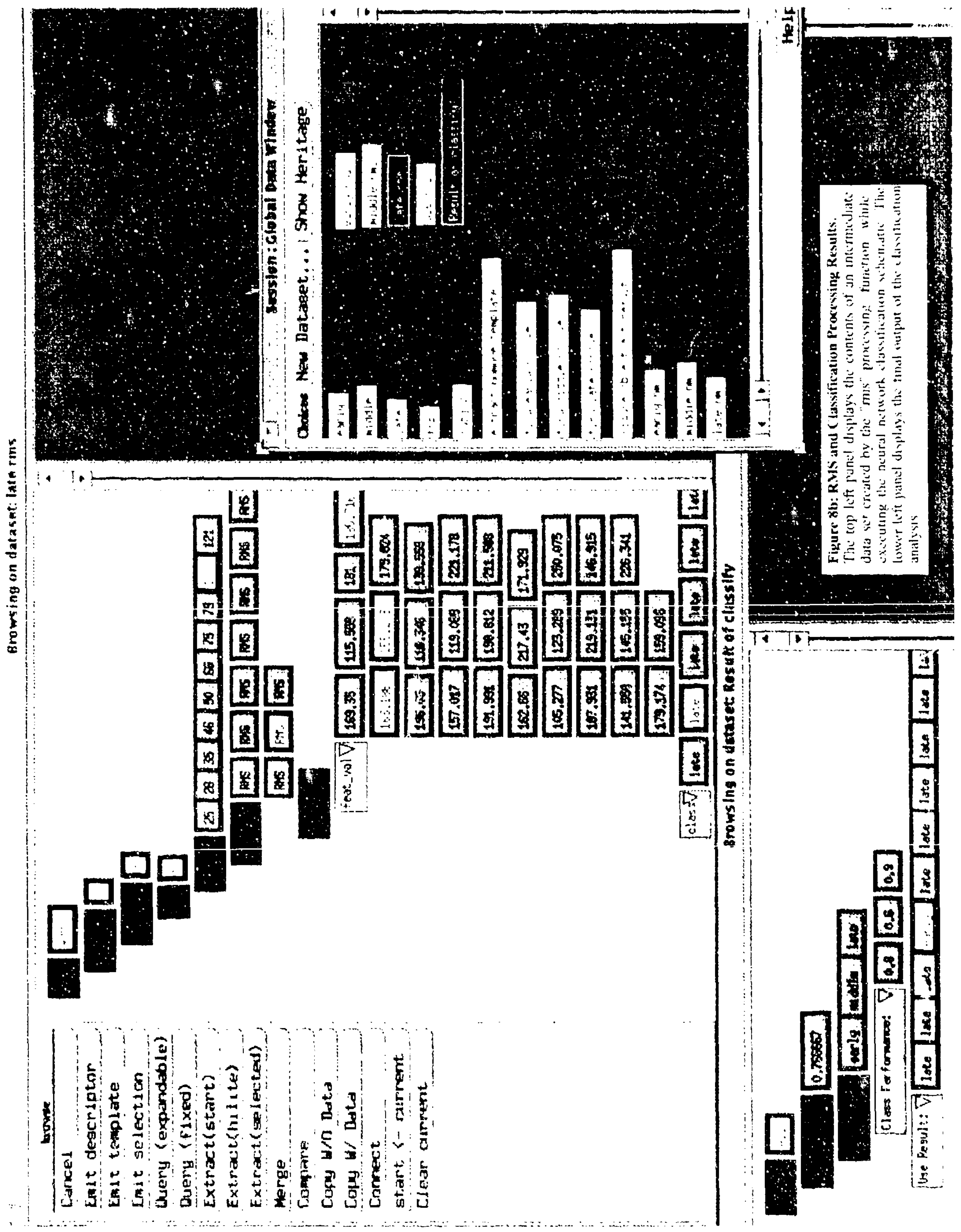


Figure 8b: RMS and Classification Processing Results.
The top left panel displays the contents of an intermediate data set created by the "rms" processing function while executing the neural network classification schematic. The lower left panel displays the final output of the classification analysis.

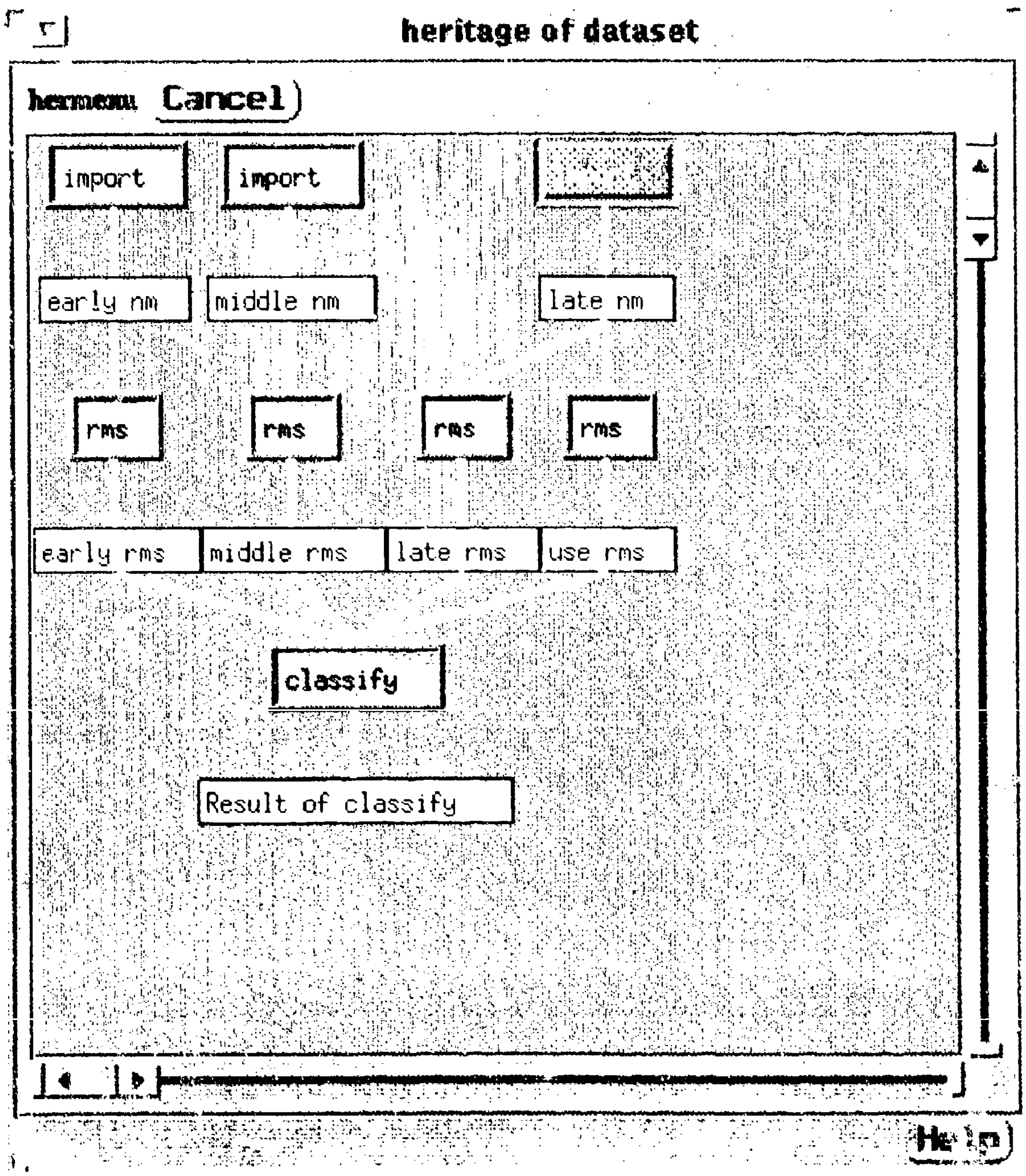
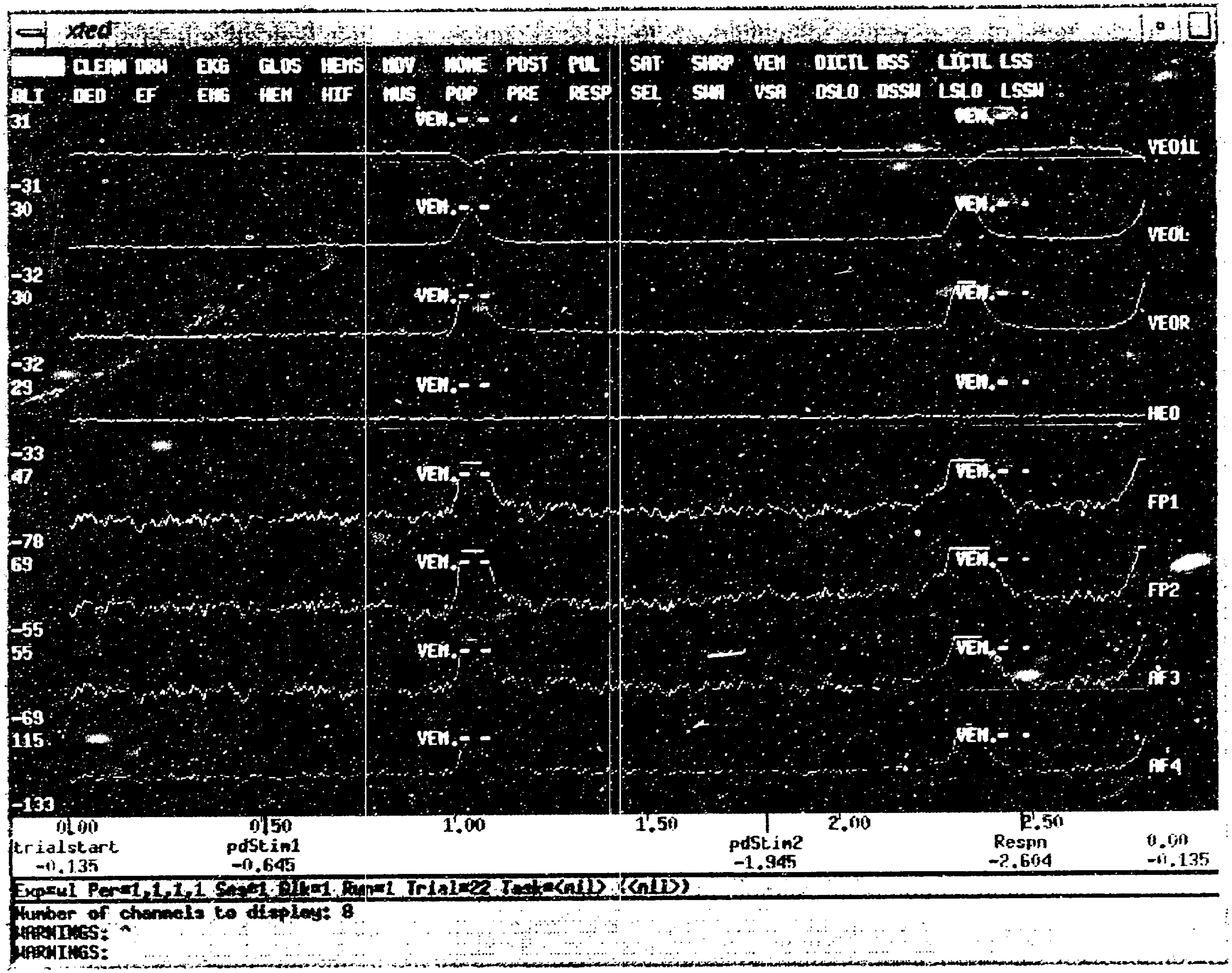


Figure 9: Heritage.
The figure displays the processing history leading up to the "Result of classify" data set. The history is automatically complied at each processing stage.

Figure 10: Screen dump from Data Reviewer ported to X-windows. The display shows data recorded during a single task trial consisting of two visual stimuli, marked by the "pdStim1" and "pdStim2" labels at the bottom of the figure, and a response, marked by the "Respn" label. Time is shown in seconds. An artifact type is selected by clicking a mouse button on one of the labels at the top. Then the artifact is selected by clicking on the beginning and end of an artifact. In the example show, the user has used the label VEM to mark out two instances of vertical eye artifact, specifically, eye blinks. The number of channels, the time and amplitude scales, and many other aspects of the display are alterable using the icons shown at the right and by commands entered through the keyboard. The data can be temporarily filtered by a wide variety of IIR filters.


xded
CLEAN DRW EKG GLOS HEMS MOV NONE POST PUL SAT SHRP VEM OICTL BSS LICTL LSS
BLI DED EF EMG HEM HIF MUS POP PRE RESP SEL SWA VSA DSLO DSSW LSLO LSSW
VEM.- -
VEOIL
VEOL
VEOR
HEO
FP1
FP2
AF3
AF4
31
-31
30
-32
30
-32
29
-33
47
-78
69
-55
55
-69
115
-133
0.00
0.50
1.00
1.50
2.00
2.50

```
trialstart      pdStim1                            pdStim2        Respn      0.00
 -0.135          -0.645                             -1.945        -2.604     -0.135
Exp=ul Per=1,1,1,1 Ses=1 Blk=1 Run=1 Trial=22 Task=<nil> (<nil>)
Number of channels to display: 8
WARNINGS: ^
WARNINGS:
```

Figure 10: Screen dump from Data Reviewer ported to X-windows.

Figure 11: Screen dump from Data Subdivider ported to Motif/X-windows.
The researcher is selecting out a subset of trials from three different experimental conditions. In each condition, subjects had performed a task requiring them to hold one, two, or three items in memory and update this memory as the task went on. The upper right panel shows the three condition names selected by the researcher ("memload1", etc.) and the interval of data during a task trial which is of interest (-0.5 to 1 second centered on "psStim2"), Data Subdivider selects that subset of trials where the electrophysiological data recorded within this interval is not contaminated by undesirable signals, such as eye-blink, as specified in Data Reviewer. It is from this subset that the data is further subdivided according to other variables as specified in the lower right panel, "RT" (reaction time), "outcome" (whether or not the subject's response was correct), and "trialtype" (in this example, this indexes the memory load required in the trial). In the three panels on the left, the researcher has selected a set of task trials that do not differ in reaction time across the three conditions. The height of a bar represents the number of observations (trials) of the class that are in the corresponding interval of the x-axis (reaction times). The tiny squares represent individua observations; their vertical positions are set so that all observations can be viewed on the plot and are otherwise unrelated to bar graph heights. Green squares indicate those trials that are included in the subset of trials while red squares indicate those that are excluded. An ANOVA is performed to compare the resulting distributions. Using this tool, a researcher can insure that the data he or she is testing are actually related to the hypothesis he or she intends and are not contaminated by unrelated sources of uncontrolled experimental variance.

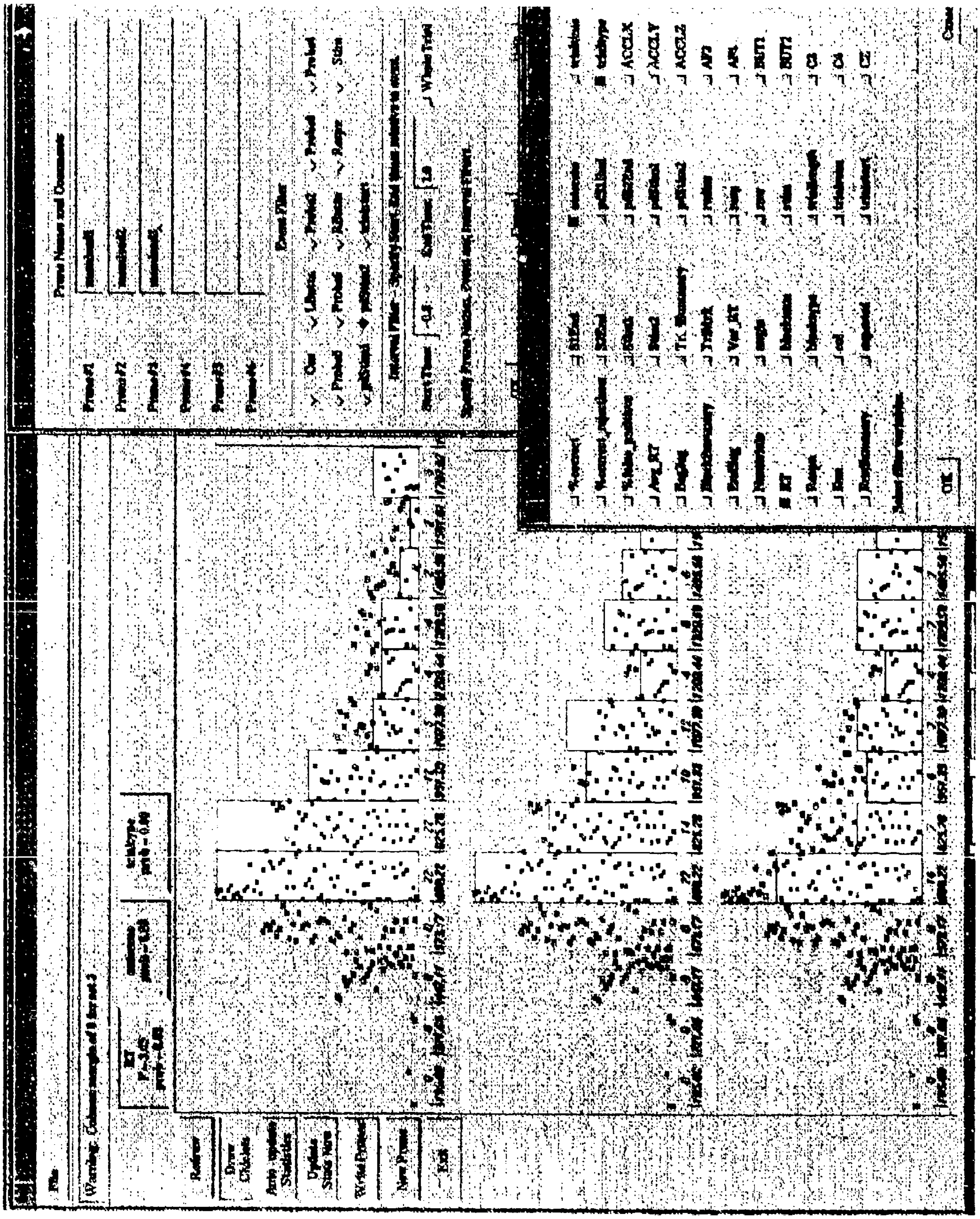


Figure 11: Screen dump from Data Subdivider ported to Motif/X-windows.

Figure 12: Screen dump from EXPLORE ported to Motif/X-windows.
The researcher is examining line plots and topographic maps of power spectra on the right and, on the left, spatially enhanced (a 3-D surface Laplacian estimate) averaged evoked brain signals time locked to "pdStim2" as specified in Data Subdivider (Figure 11). The topographic maps were derived from 32-channel recordings and data from a few of these channels are shown in the line plots; channel names appear at the top of the plots. The researcher has arranged the plots to easily compare temporal and spectral differences and spatial distributional differences of these two quantities between the three memory load conditions as described in the caption for Figure 11. The window at the upper right corner labeled "data sets" provides a means to select which data set(s) to load into a plot while the window at the lower right labeled "channels" provides a means to select which channel. The window in the center labeled "Filter" provides a selection of filters that can be reversibly applied to the data. Scaling for the topographic maps are shown in the two small windows labeled "cmap". Finally, the window at the lower left labeled "plot descriptor" shows detailed information about the data plotted in the line-plot for channel O1. The program facilitates instant creation of an analysis and display according to the immediate interests of the researcher. Many analysis functions and options, and an unlimited variety of display organizations (up to the limit of screen resolution), are available through pop-up menus.

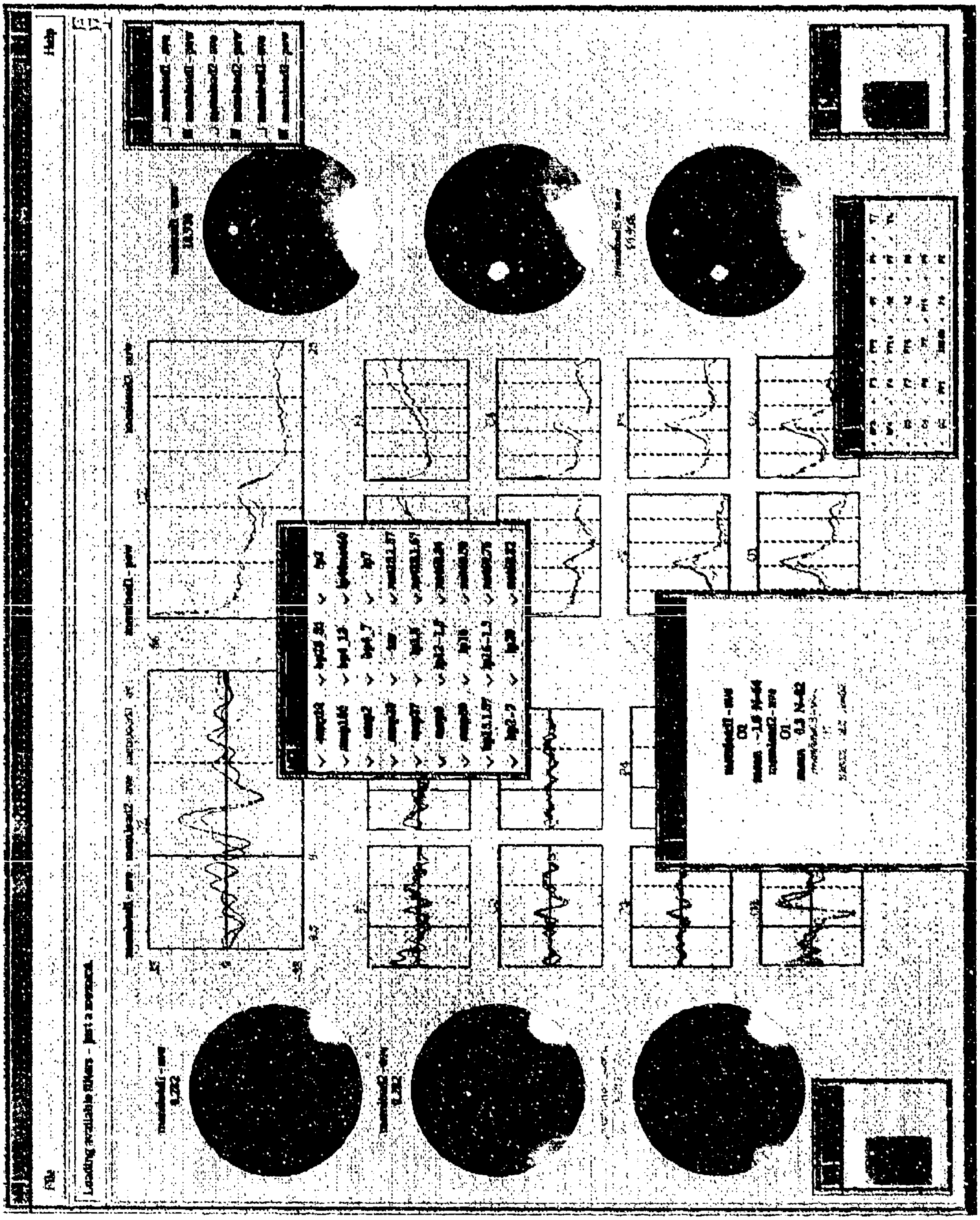


Figure 12: Screen dump from EXPLORE ported to Motif/X-windows.