# THE BROWSER PROTOTYPE

for the



# CTBT KNOWLEDGE BASE



**Hillary M. Armstrong and Ralph G. Keyser**

**Sandia National Laboratories**

**July 2, 1997**

## ABSTRACT

As part of the United States Department of Energy's (DOE) Comprehensive Test Ban Treaty (CTBT) research and development effort, a Knowledge Base is being developed. This Knowledge Base will store the regional geophysical research results as well as geographic contextual information and make this information available to the Automated Data Processing (ADP routines) as well as human analysts involved in CTBT monitoring.

This paper focuses on the initial development of a browser prototype to be used to interactively examine the contents of the CTBT Knowledge Base. The browser prototype is intended to be a research tool to experiment with different ways to display and integrate the datasets. An initial prototype version has been developed using Environmental Systems Research Incorporated's (ESRI) ARC/INFO Geographic Information System (GIS) product. The conceptual requirements, design, initial implementation, current status, and future work plans are discussed.

Keywords: ARC/INFO, ArcView, browser, CTBT, data integration, data visualization, Knowledge Base, GIS



*Sandia is a multiprogram laboratory operated by Sandia Coporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.*

## DISCLAIMER

## INTRODUCTION

The purpose of this document is to introduce the Knowledge Base Browser prototype. There are several motives behind development of a browser. Number one is the need for the various users to be able to browse the data in the Knowledge Base during all phases of development. An early benefit of this has been the realization that accurate metadata is essential to effectively work with large volumes and many different types of data. Dealing with that issue has, in turn, helped with the development of a draft data dictionary [Shepherd 1997] and associated draft dataset delivery specifications [Armstrong 1997]. Another motive is to experiment with different ways of accessing and displaying the different types of data. This will lead to discovery of optimal ways to handle each datatype. Yet another motive is to look at several commercial software packages to do the best job in developing the browser. This experimentation will point to which one is best suited to development of the browser, as well as other applications in the future. Thus, the browser prototype is an experimental tool to explore different ways to display and integrate datasets of varying type and format, residing in the Knowledge Base.

## CONCEPTUAL REQUIREMENTS

While developing an experimental prototype, it is important to keep potential functional requirements in mind to guide the development. The items in the following list describe some of the capabilities the browser prototype will need to provide. It is by no means an exhaustive list, but is a starting point. Development of the prototype will point out additional requirements. The current list reflects what potential users may be doing with the Knowledge Base. These users could be researchers, evaluators, or others who have either provided or are otherwise familiar with the data. Other types of users might be application developers who currently are or will be putting together applications which access, manipulate, or maintain the data in the Knowledge Base.

- **Allow access to and examination of the data in the Knowledge Base for such things as identification and reconciliation of disparate datasets covering the same or adjacent areas**
- **Handle large datasets quickly and efficiently**
- **Provide access to information about the datasets in the Knowledge Base (metadata)**
- **Display a wide variety of data types and formats (e.g., imagery, oracle tables, Spatial Database Engine (SDE) tables, ARC/INFO coverages)**
- **Provide access to a theoretically infinite number of datasets**

- **Facilitate user interface design discussions for other applications**
- **Have a shallow learning curve for users**
- **Be relatively easy to understand, from an application developer's point of view**
- **Be used as reference on ways to access and display data and, if appropriate, as a starting point to develop other more specialized applications, such as a maintenance interface**
- **Be composed of modular and well documented code so that pieces may be replaced and updated or used as parts for development of other applications**

## CONCEPTUAL DESIGN

The following design characteristics come out of the conceptual requirements listed above. Their proper implementation will help satisfy those requirements.

- **All datasets should be available to the user simultaneously**
  This implies that the user should be able to access any dataset from the top or major level in the application. In order to make this useful and not overwhelm the user, the datasets should be presented in a logical and organized fashion.

- **Functions that control the datasets of a datatype should be grouped together and available only at the appropriate time, and functions that apply to all datatypes in general should be available all the time**
  This implies grouping the functions such as the ones that turn specific datasets on and off, but having ones that apply to the general session environment, like redraw all, be accessible all the time. When dealing with global datasets in applications such as this, it is essential that users have control over when the datasets are drawn.

- **The user should be able to control the appearance of the data**
  The implies letting the user change the symbols and colors of features in a dataset to facilitate examination of the data.

- **The user should be able to easily and directly access the datasets' metadata and attributes**
  An online connection to metadata is one possibility here. The ability to display the attributes of selected features is essential.

- **The user should be able to move around and zoom in and out at will**
  The ability to go quickly to an area of interest is important. Also, the user needs to know ahead of time if what they are requesting will take a long time to draw.

- **The user should be able to analyze the data and then save and output these results**
  This implies provision of spatial analysis tools and hard and softcopy output tools.

- **The prototype needs to be as generic and consistent as possible in the way it accesses and handles the datasets**
  This will facilitate modifying the user interface to accommodate adding new and different types of datasets as they become available. The more commercial off the shelf (COTS) software can be used, the less "custom" the interface will be, and the easier it will be to maintain. The amount of customization necessary depends on the application software being used for development together with the user requirements.

- **The application and code should be developed and documented to some reproducible, commonly available standard**
  Doing this will make the application much easier to maintain. The application will also be more useful either as a reference for an application developer, or as a parts for a new system, since it will be easy to interpret and evaluate fitness for a specific use.

## PHYSICAL DESIGN AND IMPLEMENTATION

Since this browser is a prototype, it will continue to evolve as feedback is received from the users and new types of datasets are hooked up. It will start out to be very basic with hooks to just a few datasets and will end up being a fully functional browser with access to all the datasets in the Knowledge Base. One of the risks of this type of prototype development is that, due to lack of precise implementation requirements and scope at the beginning, the system will outgrow its original design and evolve into a kludge. That is why it is usually a bad idea to expect prototypes to be used as or evolve into production systems. One good way to help lower the above mentioned risk is to use COTS software packages as much as possible. That way, time is not spent developing basic parts that already exists commercially. This frees the developer up and allows more time to experiment with different ways of developing the unique and interesting parts of the system. Another "plus" of using COTS software is that the system will more likely to work with new releases of the COTS software package (be upwardly compatible). Although using COTS software is a good idea, it is rare to find one package that meets all development needs. When custom code does need to be developed, it should be well documented, written to a commonly available software development standard, and contain as little operating-system dependent code as possible. The more system dependent the code, the more work it will be to port the system to another hardware platform in the event that a hardware change is needed or support for an additional platform is required. These are all important considerations, because at

the initial prototyping stage, nothing has been "set in concrete" and therefore anything can change. In fact, things should change as a result of the prototyping phase if it is well executed.

**Prototype development software**

When looking at COTS development packages, the idea is to balance familiarity, capability, flexibility. ARC/INFO was chosen as the initial prototyping package both because we were familiar with it and knew it could do the job, our customer had expressed interest in it, and at the time ArcView didn't have all of functionality we were looking for. With ArcView 3.1, slated for release in "late summer 1997" [ESRI 1997], it now looks like a good choice as well. It may even have some advantages over ARC/INFO for browser prototype development. Capabilities like interfaces to handle more of the military data formats, a new image handling and analysis package called Image Analyst which was developed in close collaboration with ERDAS, Inc. (a leading image-processing software vendor), and the Dialog Designer Extension which makes it much easier to customize ArcView make it a very attracting prototyping tool.

**Browser structure**

As mentioned above, the browser is currently implemented as an ARC/INFO application. ARC/INFO is a GIS toolbox containing a wide variety of low level tools. These tools may be assembled into a collection of programs called using ESRI's custom Arc Macro Language (AML) application programming interface (API) together with menus to create a graphical user interface (GUI). ARC/INFO provides a high degree of capability as well as design flexibility. Because the API is so flexible, AMLs can become fairly complex. Though not as complex as those developed in a language such as C++ and an X windows package, the AMLs and menus must be designed to work together as a system, which requires significant investment in upfront requirements analysis and design, even in this type of prototype development.

The browser prototype needs to provide fairly rapid access to a theoretically infinite number of datasets in a variety of formats and varying sizes, and be easy to understand from an application developer's point of view. To this end, significant effort was put into the underlying structure of how the application interacts with those datasets. Managing the complexity of dealing with so many datasets ranging from raster to vector to flat files, in a variety of different formats (e.g. ARC/INFO coverages, ArcView shapefiles, oracle tables, SDE tables, Vector Product Format (VPF), ASCII, etc.), will determine how easy it is to add new datasets, change the way the dataset appeared in a display, etc. In order to help manage this complexity, each data type (e.g., event sources, meteorology, depth to moho, etc.) is loosely treated as a class of objects, with the datasets being instances of that class (i.e., objects). Although AML does not have object oriented extensions, it is possible to structure the code in a fashion that reflects this way of thinking. The assumption was made that datasets of the same type are similar enough that it makes sense to group the functions which handle them together. Handling datasets in this way has helped to manage the complexity of accessing and displaying a large number of datasets by making it easier to find and change the code that controls a particular type of data. Physically, these ideas have been implemented as described in the following paragraphs.

Datasets of each datatype are accessed via a menu like the Digital Chart of the World menu shown in Fig. 1. The menu is controlled by an AML which takes one argument. This argument is the name of the routine (a separate block of code placed at the end of the AML) to be run by the AML in order to tell the datatype to do something (e.g., initialize its menu, draw one of its datasets). These routines are analogous to the methods of an object. The routine name is also an indication of the state of the dataset. For example, when the user selects the "Digital Chart of the World" option from the "GEOGRAPHICAL" datatypes pulldown menu, dcw.aml is run with menu_init as the argument, telling the datatype to initialize and display its menu. If a dataset is then turned on and the "APPLY" button is selected, dcw.aml is called again, this time with "apply" as the argument which runs the apply routine which sets the appropriate status variables and calls the draw routine to draw the dataset which has been turned on.

Structuring the application in this way has made modifications to the code, such as adding new datasets much more straightforward. For example, to add a new dataset of an already existing datatype, add the dataset name to the global initialization routine and code specifying how the new dataset should be drawn to the global initialization and draw routines, and then add an "ON/OFF" button for the new dataset to the menu for that datatype to give the user access to the new dataset.

Another factor which has helped make development and modification easier is that the prototype has be developed following the ESRI's applications programming standards [ESRI 1992]. Among other guidelines, this standard recommends fully documenting the AMLs and menus to include a documentation header, which it provides a sample of. Although time-consuming to create, this documentation has already proven to be extremely valuable in adding new and modifying existing functionality in the prototype.

## CURRENT STATUS

A first cut of the browser prototype has been developed using ARC/INFO. It has access to 43 datasets. Some of these datasets are global and some are regional. They range in size up to 3 gigabytes. Raster and Vector datasets are represented. Basic functions, such as turning datasets on and off, changing symbol and color, and panning and zooming are available at this time. (Fig. 2)

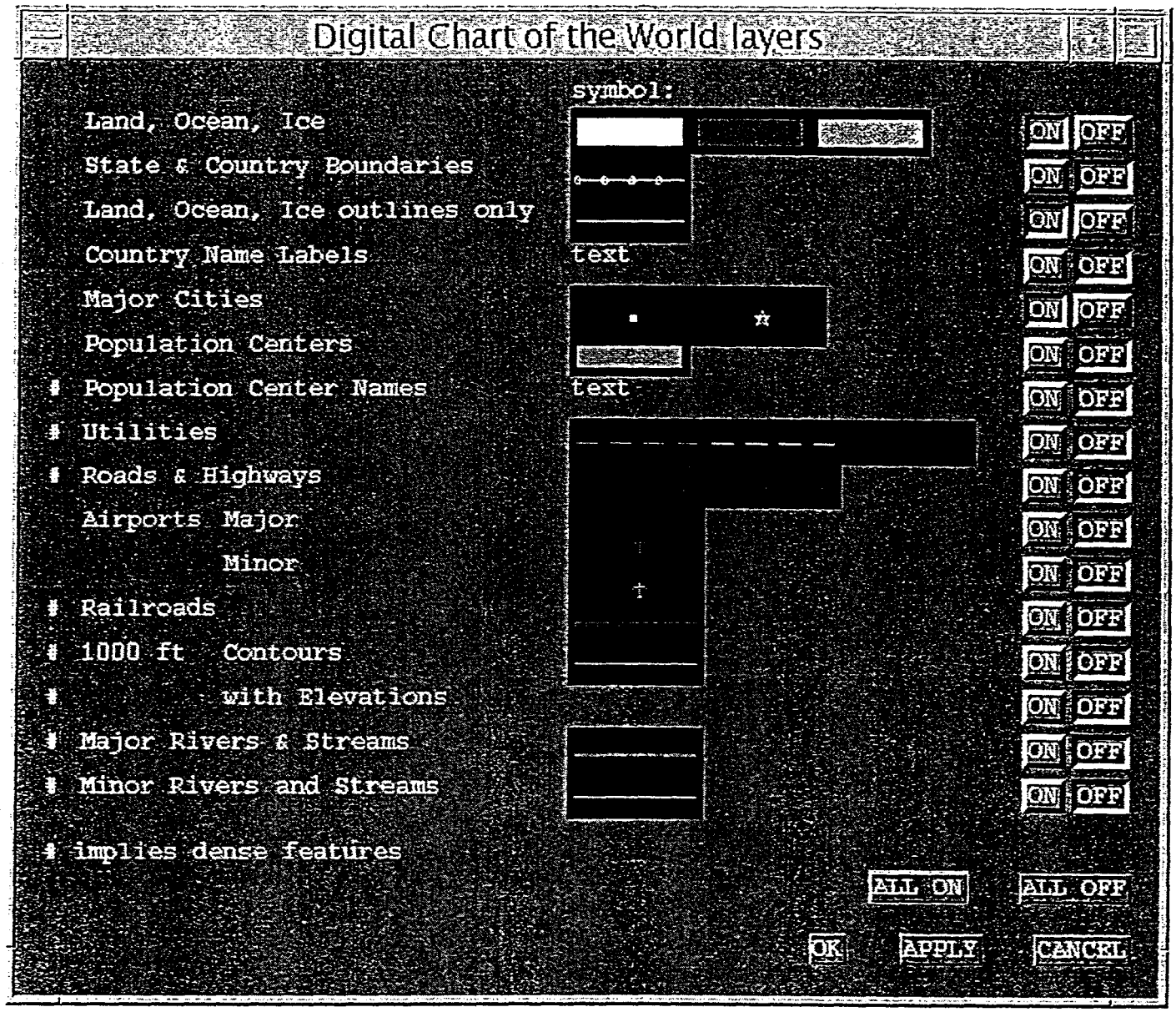


Fig. 1: The Digital Chart of the World access menu

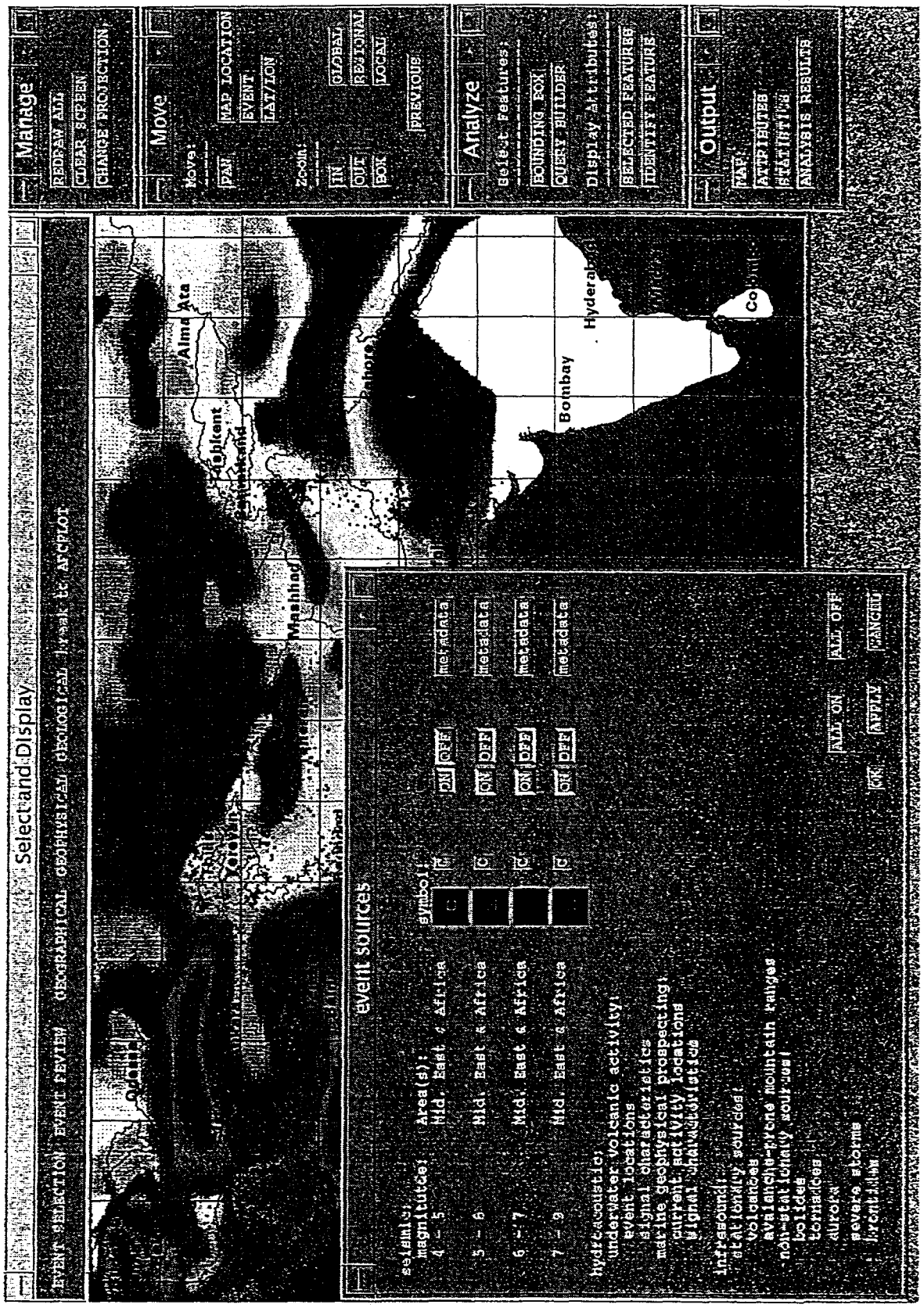


**Fig. 2: The Knowledge Base Browser Prototype interface**

## FUTURE WORK

### Experiment with ArcView 3.1

It appears that ArcView 3.1 contains some previously unavailable but needed functionality. The fact that it could be bought with ArcView and then would not have to be developed in ARC/INFO would not only save valuable development resources, but would also result in a more purely COTS application (i.e. less custom code) than ARC/INFO. Additionally, more effort could then be put into the task of developing new ways to use the existing ArcView tools to access and display the data, rather than developing the tools themselves. The generic GUI which come with ArcView could be used first, and then gradually customized to meet the users' needs. In contrast, even the most generic ARC/INFO interface will be custom to some degree, because of the nature of ARC/INFO application development. A big tradeoff here would be that the generic interface from ArcView would probably not be as easy to use as a custom one developed with ARC/INFO, but it could be had much faster and may be suitable as a prototype. Thus, it would be worth the effort to look into using ArcView to develop the next version of the browser prototype.

### Hook up more datasets

As the research entities deliver more and more datasets for inclusion in the Knowledge Base, they will be hooked up to the browser. This will be an ongoing effort. During this period, the suitability of the metadata delivered with the datasets will be evaluated, as it will be used to install the datasets in the Knowledge Base. Eventually, users will have access to spatial datasets like bathymetry, seismic events, correction surfaces, hydroacoustic event, to name a few.

### Demo to users and refine user interface

Development of the browser has been and will continue to be user-driven. Thus, we will continue to periodically give demonstrations to and solicit comments from the users. Comments and questions will continue to influence things like the look, feel and features of the browser.

## References

Shepherd, E., R. Keyser, H. Armstrong, E. Chael, C. Young. Draft Data Dictionary for CTBT Knowledge Base, Revision 1. April 21, 1997.

Armstrong, H., R. Keyser. ARC/INFO Dataset Delivery Specifications. February 3, 1997.

Environment Systems Research Incorporated. Arc News. Spring 1997.

Environment Systems Research Incorporated. Applications Programming Coding Standards. ESRI Applications, March 1992.