The Load Data Analysis Workstation (LDAW) is a SAS/AF® information system application designed to allow electric utility staff to access, analyze, and present results from a variety of databases. Developed by Quantum Consulting Inc. (QC) for the Electric Power Research Institute (EPRI), LDAW version 1.0 runs under SAS 6.04 for MS-DOS, and has been available to EPRI member utilities since June, 1991. The LDAW is still under active development. Version 1.2 will contain several enhancements, and is scheduled to be released by June, 1992. A UNIX version of LDAW 1.2 for UNIX C will also be released later this year.

The LDAW uses a SAS/AF user interface to combine several SAS® packages into a coherent system. The SAS/AF code controls user input to over 100 individual SAS programs, which make use of BASE SAS®, SAS/GRAPH®, SAS/STAT®, and SAS/FSP®. The user interface allows consistent input and output protocols for an otherwise unwieldy library of programs.

This paper is the second SUGI presentation to address the LDAW project. At SUGI 16 in New Orleans, we presented a paper that emphasized the analysis capabilities of the LDAW system. This paper, for SUGI 17 in Honolulu, will discuss the merits of the SAS/FSP®. The user interface allows consistent input and output of data on a variety of hardware platforms.

"Knowledge itself is power" — Sir Francis Bacon

The Information Age has resulted in the collection of large amounts of data regarding a wide variety of phenomena. In the utility industry, the enormous amount of data collected annually can provide valuable knowledge and insights into the nature of complex processes. However, knowledge requires analysis. Huge volumes of data are overwhelming and meaningless without analysis techniques that summarize, consolidate, and clarify data into comprehensible results. It is impossible to draw reliable conclusions by looking at just a few data points, in the same way that it is difficult to characterize a forest by looking at a single tree. Clearly, tools and techniques are required to make sense of the vast amount of information gathered in any data collection effort, and thereby to provide knowledge.

The Problem — Making Sense of Large Amounts of Data

The LDAW project is concerned with providing knowledge and insights about one subject in particular: customer electricity usage patterns. Electric utilities collect large quantities of data in order to achieve this goal. The data are used in a variety of analysis and forecasting activities. These include:

- Rate Design
- Forecasting
- Marketing
- Demand-Side Management
- Fulfillment of Governmental Regulations

The collection and analysis of detailed customer electric usage data is generally referred to as load research. A utility load research department usually collects data from customers in the residential, commercial, and industrial sectors of its service territory. Load research data include time series kWh data, weather data, survey data, and engineering measurements. An ambitious load research program involves the collection of data from several hundred sites in each of the three sectors. Data are gathered by scientifically selecting a sample of customers in a given sector and installing metering devices that record the energy usage of either an entire building, or of a given appliance or "end use." Usage is recorded at frequent intervals, typically every 15 minutes. Gathered data are downloaded by modem or hand-held reader to a central computer system for analysis.

Although most utilities perform load research in order to reveal insights about their customers, the 1978 Public Utilities Regulatory Policies Act (PURPA) provides an additional incentive to collect load data. This Federal law requires utilities to meter whole premise electricity usage for a sample of their customers, in order to establish the prices that they will pay for power from non-utility generators. As a result of the PURPA requirements, almost every large utility performs some sort of load research.

The size of load research databases can be quite large — up to several hundred megabytes of primary data per year. Drawing accurate conclusions from this data is a challenging data analysis problem.

The Solution — LDAW, a SAS/AF Information System

The collection and analysis of load research data is not a new field. Load research data have been collected and analyzed on mainframe computers since the 1970's. However, at many utilities, mainframe data collection and analysis is expensive and inefficient. The LDAW takes advantage of the recent advancements in PC and UNIX workstation technology in order to provide a modern, cost-effective alternative to outdated and expensive mainframe load research systems. In addition, the LDAW incorporates state-of-the-art analysis procedures that provide a new level of performance and reliability.

Most utility load research departments currently use SAS in a mainframe environment for processing. As a result, utility staff members are familiar with SAS, and a considerable amount of development time has already been spent creating mainframe SAS programs to perform various load research tasks. The SAS system was chosen as the development platform for the LDAW because it is the standard statistics package for the utility industry. The LDAW combines the familiarity of SAS with the advantages and flexibility of a personal workstation environment. The LDAW was designed with an open architecture in order to tap into the existing knowledge base of mainframe SAS programs. Virtually any existing mainframe SAS program can be easily modified and installed within the LDAW system.

LDAW System Design

The LDAW consists of a SAS/AF user interface that serves as a front-end to SAS analysis modules. The user interface is written in Screen Control Language (SCL), and consists of a menu system that accesses program input screens. The program input screens help the user select valid input parameters by providing selection lists, pop-up menus, and help screens. Associated with each program input screen is a SAS analysis module that actually performs the given task.

The design of the LDAW reflects the manner in which it has been created. Originally, the LDAW "system" consisted of a set of roughly 50 separate SAS programs that performed data input, validation, analysis, and presentation. This system was difficult to use, because each program had to be painstakingly changed by a
SAS programmer every time new data were analyzed. In order to improve the system, the programs were rewritten so that all relevant parameters could be easily changed through macro variable referencing. A SAS/AF interface was then created as a user friendly front-end to the SAS programs. This allows the programs to be run by users with only a minimal knowledge of SAS.

User Interface

The LDAW uses an innovative menu system that is similar in appearance and function to the PROC PMENU menu system available for SAS 6.06. However, PROC PMENU is not implemented for the current release of SAS for the PC, and the LDAW menu system is still superior to anything currently available for the PC. The LDAW menu system, described at length in the SUGI 16 paper, arranges main menu options along a bar that appears at the top of the screen. The main menu options access submenus that group together programs that perform common tasks, such as data input, data validation, data analysis, or graphing. The main menu options are ordered in a manner that visually reinforces the logical progression of the tasks to be performed. Selecting a main menu option pulls down a submenu listing available programs. An example of the LDAW menu system is shown in Figure 1. The cursor is moved using the <TAB> key. An option is selected from a menu by moving the cursor to that option, pressing <ENTER> once to highlight, and then again to select. A mouse can also be used to move the cursor and select options, by clicking the left mouse button once to highlight an option, and then again to select.

![LDAW Menu System](image1)

Selecting an item from a submenu accesses a program input screen. An example of an LDAW program input screen is shown in Figure 2. Numerous bells and whistles provide useful error trapping for invalid input parameters.

![LDAW Program Input Screen](image2)

Analysis Modules

SAS analysis modules perform a variety of data input, validation, analysis, reporting, and graphing tasks, as summarized in Figure 3.

![Summary of LDAW Analysis Capabilities](image3)

Input

- ASCII Flat File
- Lotus® File
- dBASE® File
- EPRI Format Files

Validation

- Spike Detection
- Cross Validation
- Data Editing
- Descriptive Statistics

Analysis

- Average Load Profiles
- Summary Statistics
- Load Duration
- DSM Evaluation
- Daytype Analysis

Output

- Raw Data Reports
- Average Data Reports
- ASCII Flat File
- Lotus® File
- dBASE® File
- EPRI Format Files

Graphing

- Load Profiles
- Duty Cycle Histograms
- Connected Loads
- Summary Statistics
- 3D Surface Graphs

Data input takes place through import modules that convert data in Lotus®, dBASE®, or EPRI formats into a SAS dataset. The LDAW can also convert ASCII files in a wide variety of formats into a SAS dataset. If the user already has data in a SAS dataset, then an LDAW convention called a data description can be quite useful. A data description maps the names of variables in a user's SAS dataset into global macro variables that are referenced in every LDAW SAS module. The data description convention provides the LDAW with the flexibility to adapt to a wide variety of variable naming conventions, rather than being "hard-wired" for one type of data.

Data validation modules clean and validate input datasets. The Spike Detection module detects false data readings, while the Channel Cross Validation module compares whole premise load versus the sum of end-use load. Data can be manually edited, using a module that calls PROC FSEDIT. A broad range of SAS procedures can also be called to provide descriptive statistics.
Data analysis modules provide the bulk of the power of the LDAW. Average Load Profile analysis develops mean average load profiles by selected variables. The Summary Statistics module provides the load factor, diversity factor, and Unit Energy Consumption for each channel. The DSM Impact Evaluation modules determine the impact from hypothetical load control programs.

LDAW can also create a variety of output. Data reporting modules produce detailed reports listing the output of LDAW analysis modules. Data can be output in a raw form for each channel, or average load data can be output. Data can also be exported to a variety of formats.

LDAW graphing modules provide a quick way to visualize raw data, or to graph the results of other analysis programs. Output can be routed to any valid SAS device, or to a Graphics Stream File (GSF).

When a program is executed, all SAS/AF input parameters from the program screen are passed to macro variables, and the external program is called using a %include statement. This takes place in an SCL submit block, which is a facility for interfacing compiled SAS/AF code with interpreted SAS statements. Calling the program with the %include statement allows the entire program to physically reside outside of the LDAW SAS/AF catalog. This means that the analysis code can be modified without recompiling the interface code. The external program contains references to the macro variables representing user input. The external program also contains references to the macro variables defined in the data description.

Program execution takes place in either Immediate or Batch mode. Batch mode creates a SAS program and writes it to an external file for later execution. Immediate mode creates a SAS program and executes it immediately.

The Workstation Platform

The LDAW was developed for PC and UNIX workstation platforms, because of numerous advantages over the mainframe, as summarized in Figure 4. The mainframe computing environment is very expensive, and can be difficult and costly to maintain. Mainframe operating systems such as VMS, CMS, and TSO are difficult to use, and the user interface is poor. Most mainframe users do not have access to good on-screen graphics. Speed and performance on a mainframe depend on the number of users and the user priority. Although mainframes do have lots of storage capacity, most desired data is not kept on-line. Mainframe systems have lots of memory and good networking capabilities. However, mainframe data analysis systems are often bogged down by the large volume of jobs being processed throughout the utility. The user is also left at the mercy of an MIS department in order to get data and upgrade equipment.

In contrast, PC and UNIX platforms leave the user completely in control of data and equipment. Recent advances in hard disk technology enable workstations to access several gigabytes of data. The workstation platforms are also inexpensive, easy to use, and allow the user to maintain control over data and equipment. Networking capabilities are also present. Data visualization and graphics are good on a PC, and outstanding on a UNIX workstation.

Mainframe advocates often argue that workstations are not fast enough, and cannot provide access to all users on a system-wide basis. While advances in workstation technology such as improved processing power and improved networking will eventually solve most of these problems, mainframe advocates see this as re-inventing the wheel. A powerful networked workstation environment will simply be an inferior mainframe. This common argument effectively ignores the fact that a real revolution has occurred in data processing. While mainframe systems will probably always retain a role for very large scale processing or centralized data storage and access, PCs and UNIX workstations will continue to take over more "mainframe-like" activities because they are cost-effective, easy to use, and let the user maintain control over data and equipment. In addition, PCs and UNIX workstations are designed using modern computing principles and provide access to the latest advances in software engineering, rather than the often outdated and inferior software packages found on mainframe systems.

Figure 4

Comparison of Processing Environments

<table>
<thead>
<tr>
<th>Graphics</th>
<th>Mainframe</th>
<th>PC</th>
<th>UNIX Workstation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Poor</td>
<td>OK</td>
<td>Excellent</td>
<td></td>
</tr>
<tr>
<td>Speed</td>
<td>Variable</td>
<td>Moderate</td>
<td>Fast</td>
</tr>
<tr>
<td>Ease of Use</td>
<td>Cryptic, JCL</td>
<td>Easy</td>
<td>Variable</td>
</tr>
<tr>
<td>User Interface</td>
<td>Poor</td>
<td>Better</td>
<td>Excellent</td>
</tr>
<tr>
<td>Storage</td>
<td>Lots</td>
<td>Less</td>
<td>Even More</td>
</tr>
<tr>
<td>Cost</td>
<td>Very Expensive</td>
<td>Cheap</td>
<td>Moderate</td>
</tr>
<tr>
<td>Maintenance</td>
<td>Expensive</td>
<td>Cheap</td>
<td>Moderate</td>
</tr>
<tr>
<td>Memory</td>
<td>Lots</td>
<td>640 K</td>
<td>lots</td>
</tr>
<tr>
<td>Multi-User</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Multi-Tasking</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Networking</td>
<td>Variable</td>
<td>OK</td>
<td>Good</td>
</tr>
<tr>
<td>Control</td>
<td>MIS Department</td>
<td>You</td>
<td>You</td>
</tr>
</tbody>
</table>

A further abstraction of the relationship between PCs, UNIX workstations, and mainframes is found in Figure 5. This simplex graph portrays how each platform fits into the interaction between technology, time, and effort. The three vertices of the triangle are labeled as technology, time, and effort nodes. The technology node represents the level of technology required by the platform. The effort node represents the amount of manpower and effort required to create, implement, use, and maintain the platform. The time node represents the amount of time required by the platform. The vertex of each node is the point at which each node is at a minimal value. Moving a point away from a vertex increases the value of that node. Note that in order to minimize one particular node, one must maximize the other two nodes. For example, for a solution to minimize time, it must maximize energy and technology. Cost is a non-linear function of all three nodes. Although appealing philosophically, it is usually not cost effective to completely minimize one node. While real world solutions often strive to minimize one node, in practice they end up finding a balance point between the three nodes.

In Figure 5, the mainframe, workstation, and PC positions indicate the approximate balance points of each computing strategy. For example, a mainframe uses a good deal of technology and effort in order to minimize time. However, all of the technology and effort required result in a very expensive solution. In contrast, a PC uses a good deal of time and technology in order to reduce the cost. The cost of this solution is very reasonable. The UNIX workstation provides the most evenly balanced solution between the three nodes, at a price only slightly above the PC.
Figure 5
Attributes of Four Possible Load Research Platforms

Mainframe: Low time value, but high technology and effort

UNIX Workstation: Medium effort, technology and time

PC: Requires less effort and technology, but more time

Abacus: Requires minimal technology, but high effort and time
While PCs and workstations provide the best performance value, all three solutions are preferable to the fourth possible strategy: an army of people with abacuses assembled to perform all data processing. This solution would minimize technology at the expense of a maximum of time and effort!

LDAW Design Features

The LDAW system contains over 50,000 lines of code. Simply maintaining and debugging this code is a large job. However, the job has been made easier and the number of bugs minimized by observance of several design features. These design features have contributed to the development of robust code, and have also enabled the project to be easily expanded. Figure 6 presents a summary of LDAW design features.

Figure 6
LDAW Design Features

A strategy for a robust application

- Modular Design
- Divide and Conquer
- SCL Macros
- External Analysis Programs
- Standardized Code Look
- Expandable Loose Leaf User's Guide
- Frequent Comments and Input From LDAW Users

**Modular Design**

The most important design feature is the modular design of LDAW programs. By separating the user interface from the analysis modules, SAS/AF programmers can develop the user interface while SAS programmers develop analysis modules. This division of labor has allowed staff strengths to be concentrated. The user interface can be fully tested without having to actually run the lengthy SAS analysis modules. In addition, the analysis modules can be developed by staff that are not familiar with SAS/AF or SCL.

**Divide and Conquer**

The modular design philosophy has been used to develop analysis programs that stand alone, and require a minimum of interaction with the rest of the system. The LDAW uses a "divide and conquer strategy" to achieve this goal. By dividing the entire system into separate modules, development can proceed in steps, with essential programs being completed first. The LDAW menu system serves to integrate the independent programs into a coherent system. However, since the programs are independent, a change in one program does not require changes throughout the rest of the system. By avoiding this "ripple" effect, development can proceed at a rapid pace, without adverse effects to other parts of the system.

**SCL Macros**

The SAS/AF user interface takes advantage of compiled SCL macros. An SCL macro is a macro that contains a group of SCL statements. The SCL macro generates the SCL statements when the macro is invoked at compilation time. Since LDAW contains over 100 program screens, the SCL macros provide a way to standardize frequently used routines, and operate similar to the way function calls are used in C or PASCAL. When a bug is found, it can often be fixed within an SCL macro, thus providing a convenient way to fix the same bug in every program. In addition, SCL macros provide a way to disseminate standardized messages and warnings that can easily be changed.

**External Analysis Modules**

Since all analysis modules are physically kept in separate files outside of the user interface catalog, analysis module code can be modified without having to recompile the user interface code. This provides a very good development and debugging environment. In addition, analysis module code can be easily customized. Although SAS is often criticized for being an interpreted language, it is this very feature that is a powerful asset. The LDAW is not a black box consisting only of executable code. Rather, all the analysis source code is fully available for examination, documentation, and easy customization.

**Standardized Code Look and Structure**

Any large programming project will involve many programmers who have different styles. All LDAW code is written to conform to a standard look and structure. As a result, the code for the entire system is similar, and is easily readable and comprehensible. Attempts have been made to provide standard variable names, data stop names, SCL labels, macro variable names, comments, and documentation. In addition, each program contains a detailed header that lists the programmers that have worked on a program, the dates the program has been modified, as well as notes on how the program works. Although painstaking, a standardized code look and structure is essential to the success of a long term software development effort. Over the course of several years of LDAW development, at least 20 people have been involved in writing code. If each programmer had written the code to his own specification, the result would be an incomprehensible Tower of Babel.

**Loose Leaf Format User's Guide**

Since the LDAW is still under active development, the LDAW user's guide has been kept in a loose leaf three ring binder format. This provides for easy addition of new material as the system expands. The loose leaf format also allows updates on program operation, and correction of typos and errors.

**Frequent Comments and Input From LDAW Users**

The progress of LDAW work is frequently evaluated through regularly scheduled user's group meetings and forums. In addition, users often call with comments, or leave electronic mail on EPRINET, a world-wide network that provides support for EPRI products. Such commentary is invaluable in the creation of a reliable and well-designed product.

**Conclusion**

The LDAW is still under active development. Version 1.2 will contain several enhancements, and is scheduled to be released by June, 1992. A UNIX version of this software will also be released later this year. Version 2.0 will contain many major new features, and will run under SAS 6.07 on a variety of platforms, including Microsoft Windows.

Any questions about the LDAW should be addressed to either John Powers or Burtt Blodgett at:

Quantum Consulting Inc.
2030 Addison Street, Suite 410
Berkeley, California 94704
Telephone (510) 540-7200