ABSTRACT

The capabilities of the Statistical Analysis System (SAS) are much greater than simply providing statistical analysis. At the North Central Texas Council of Governments (NCTCOG), SAS is used primarily for its data management and programming capabilities. The use of SAS has significantly reduced the programming time to respond to ad hoc requests and has consequently changed the way in which data processing support is provided at NCTCOG. This paper will outline the process of selecting SAS, provide a comparison of SAS versus other programming languages from the standpoint of program development cost and processing efficiency, and discuss the cost-savings of using SAS.

INTRODUCTION

Although the original intent was to obtain a database management system (DBMS), SAS was chosen over four database managers as the system that would provide support to a newly developed fast-response transportation planning methodology. The cost and user interface aspects of the selection process will be discussed. Since its installation in the Transportation and Energy Department of the North Central Texas Council of Governments one and one-half years ago, SAS has become the most used programming language among the 12 data processing users in the department, and, in fact, is used more frequently than any other computer program.

The primary purpose of SAS was to allow the user to associate and query the more than 20 data files of the newly developed transportation planning methodology. The strong use of this query capability is evidenced by the short average development time for a SAS program of 30 minutes. However, SAS also may be used for larger development projects, including an accounting and performance reporting system for summarizing usage of all computing resources at a remote computing service vendor, and a dataset description system for reporting user-prepared descriptions of all tape and disk datasets.

SAS has greatly increased the amount of work that can be accomplished by nonprogrammers and has effectively increased the size of the programming staff by shifting the development of much ad hoc software from the programmer back to the end user. This provides a conservative savings of one programmer per year.

WHY A DBMS?

In 1976, development of a new automated transportation planning process was begun. This process utilizes up to 20 interdependent data files which must be coordinated and maintained concurrently. Although there is a certain amount of data redundancy in the files, data redundancy was minimized to provide for more straight-forward updating procedures, less storage cost, and faster processing times. However, for ad hoc data analysis, several of the files must often be combined to obtain the needed data.

Although the staff had not used a database management system before, it was felt that a database manager was needed to reduce the amount of software development required to handle the files and to reduce the operational complexity of merging the files as needed.

The desired benefits from a database management system were to:

- Integrate data files
- Minimize data redundancies
- Minimize required user knowledge about file structures
- Reduce I/O and storage costs

Several database management systems were investigated, and their representative costs for a monthly lease (in 1977) are shown in Figure 1, along with the cost of the SAS package that was ultimately acquired. The cost for the packages includes any query language that was available and an on-line interface. The query languages provided were primarily oriented toward simple record retrieval and report formatting. Three of the DBMSs investigated provided arithmetic operations; however, this was limited to little more than the four basic arithmetic operators. Even with a query language, the database management systems investigated would have required a heavy support function to provide the needed record manipulations and reporting.

FIGURE 1

COMPARISON OF DBMS COST VS SAS

Even though a database management system would have required a programmer to accomplish most of the system interaction, the data base approach was rejected primarily because of the high cost of the software relative to the total computer cost of the transportation planning process ($65,000 in 1978). The continued search for other options led to the analysis of statistical packages.
and the selection of SAS (based primarily on the available data management capabilities). The cost of SAS fit easily into the project budget and saved a potentially large cost for developing the needed capabilities in-house since the yearly lease cost for SAS would pay for only two weeks of a programmer's time. The use of SAS also represented an initial cost-savings ratio of 23 to 1, compared to the least expensive data base management system investigated.

After the acquisition of SAS, the realized benefits included the following:

- Integration of data files through merges
- Data redundancies controlled by ease of merge
- Necessary user knowledge of formats reduced through stored macros
- Reduction of cost through off-line storage
- Reduction of program development time
- Complete statistics package
- Powerful query capability
- Programming accomplished by nonprogrammers

All of the desired benefits of a DBMS were obtained in an acceptable way through SAS. Since integration of data files was a requirement in developing a user-oriented process, the SAS merge capability provided this basic requirement. Minimizing user knowledge of file formats is accomplished by coding SAS macros for all input and output functions. Users are required only to know the file type in which they are interested and the mnemonic names of data items.

However, the last two capabilities in the above list (which were not required of a DBMS) have represented the greatest impact at NCTCOG. By using SAS, the programming burden that would have been created by the data base managers investigated has been removed from the programming staff and is accomplished by individual users. The cost impact of this change is explored in the last section.

SAS VERSUS OTHER LANGUAGES

A comparison of SAS with other programming languages will show why SAS can have a significant impact on providing data processing support.

A sales brochure has reported program development times to prepare the same sales report in COBOL, FORTRAN, BASIC, and a query language, as shown in Figure 2. This figure shows that the query language (not SAS) could save up to 90 percent of the program development time required to prepare this particular report. This is explained, in part, because the "data management functions of collection, verification, storage, retrieval, and maintenance of data have been estimated to account for 40 to 60 percent of the programming effort required to implement an applications program." This example was admittedly aimed at the capabilities of the particular query language; however, local experience with SAS tends to support the general relationships shown in Figure 2. For example, the development of a mailing list maintenance system was recently considered in both SAS and COBOL. The cost of developing the system in COBOL was estimated, by a person unfamiliar with SAS, to be $2,000. The actual cost of the implementation in SAS was $40 for computer time, including the first production run, and two days of programmer effort, for a total cost of $200.

FIGURE 2

PROGRAM DEVELOPMENT TIMES

SALES REPORT

LANGUAGE LEVEL

The concept of language level can also be used to compare SAS with other programming languages. Language level is used to show increasing power of the programming language with increasing values of language level. The higher the level of language, the more quickly or easily a concept may be grasped by a person who understands the language. Additionally, the effort required to program in any given language follows a relationship of the squared inverse of the language level. The language level for several programming languages has been calculated by Holsteed. These, along with the calculated value for SAS, are shown in Table 1. Since language level is computed directly from an implemented program, the largest value of standard deviation for SAS is due, in part, to the fact that a SAS program may contain very high-level procedure calls and very low-level arithmetic statements. The high-level procedure calls are the main reason behind the high language level for SAS. In using these procedures, it is only necessary to understand the concept of each capability and not the technical details of how it is accomplished. These concepts may be combined and implemented much more quickly in SAS than in lower-level languages. For example, the statement PROC PRINT is used to produce the listing shown in Table 2. Any programmer knows the difficulty in producing such a generalized output. In addition, it is much easier to identify and correct a logical flaw at this level than at a more detailed level.
A more rational approach would be to select a language that is specifically oriented to an application area and provide a somewhat less powerful programming capability for developing applications outside of that specific application area. The terminology, or syntax, of the language should be oriented toward the intended user group. The correctness of this idea is shown by the fact that there are currently over 167 different programming languages in use today.

Still, a fundamental characteristic of a language is the amount of time required to be able to use that language effectively. Figure 3 shows the observed SAS learning curve at NCTCOG as reported in the user survey. The usage threshold as defined in this figure represents the point at which useful programs can be implemented in the language. A most important characteristic of SAS indicated by this curve is that functional SAS application programs can be executed at a very early point on the learning curve. Since our use of SAS is primarily in the data management area, this point tends to be at a 5-10 percent level of SAS knowledge, requiring an average of two days of learning effort. The execution of statistical procedures, without great use of data management capabilities, should require even a reduced level of SAS knowledge and learning time.

**TABLE 1**

<table>
<thead>
<tr>
<th>Language Level</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>English</td>
<td>2.16</td>
</tr>
<tr>
<td>SAS</td>
<td>2.06</td>
</tr>
<tr>
<td>PL/I</td>
<td>1.53</td>
</tr>
<tr>
<td>ALGOL 58</td>
<td>1.21</td>
</tr>
<tr>
<td>FORTRAN</td>
<td>1.14</td>
</tr>
<tr>
<td>Assembly</td>
<td>0.88</td>
</tr>
</tbody>
</table>

Although much more research must still be done on the concept of language level, the current calculations indicate that SAS provides a powerful programming tool and should significantly reduce programming time as compared to FORTRAN. At NCTCOG, the development time for SAS programs ranges from a few minutes to several days. In a recent survey, NCTCOG users reported an average development time of 30 minutes for a SAS program (comparing favorably to the time reported in Figure 2). The largest development effort has been an accounting and performance reporting system that required approximately eight days of programming effort.

**LEARNING CURVE**

In addition to reduced time and effort in programming made possible by a given language, it is important to examine the learning curve for acquiring the basic skills to use that language. As a programming language becomes more powerful, the number of available operators in that language increases until, at the extreme, there would be a single operator to perform any desired function. This implies an infinite number of operators requiring an infinite amount of time to learn the language. A more rational approach would be to select a language that is specifically oriented to an application area and provide a somewhat less powerful programming capability.
Although this curve has not been compared with learning curves for other languages, it seems logical to assume that procedural languages such as FORTRAN or COBOL would require a much greater amount of time for the learning curve to reach the usage threshold. Additionally, the greater number of unique operations required to implement a useful program in these languages should move the usage threshold up on the y-axis.

The curve also shows that SAS knowledge is acquired very quickly within the first two weeks of using SAS, and that less than half of the capabilities of SAS is used by the average user. In fact, only one user reported using more than 50 percent of the capabilities of SAS. This is due to the fact that statistical procedures, other than descriptive statistics, are used very rarely.

LEARNING TOOLS

The SAS Introductory Guide has been used as a learning tool by nonprogrammers, with good results. However, programmers and some nonprogrammers still use the SAS Users Guide as the basic learning tool. There seems to be little difference in the learning curve using these two approaches. In the small sample available, this may indicate that the Introductory guide increased the learning speed of users who otherwise would not have grasped the basic concepts so quickly. One additional learning aid, a SAS reference card, has been developed to provide a one-page overview to several of the most often used functions in SAS. This tool is used after the basic SAS knowledge has been acquired to speed the development of individual applications.

Although no quantitative statistics are available, our experience seems to show that the relative use of SAS capabilities (in order of decreasing use) is as follows:

- Data management
- Proc Sort
- Proc Print
- Proc Freq

PROCESSING EFFICIENCY

The processing efficiency of SAS versus other languages has been compared only for record I/O time. The arithmetic processing of SAS is intuitively felt to be similar to other programming languages and, therefore, has not been investigated further.

A comparison of the relative read times in SAS is shown in Table 3. This comparison is for a dataset containing 10,000 records, 165 bytes long. SAS is shown to be equivalent to FORTRAN for card image reading and approximately 50 percent slower than FORTRAN for binary data transfers. The processing cost for reading the EBCDIC dataset in SAS was $20. This has not had an adverse impact on our use of SAS to this point, although the cost of reading the large datasets (10,000 records) used in current applications is considered in the development of SAS programs.

<table>
<thead>
<tr>
<th></th>
<th>EBCDIC</th>
<th>BINARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAS</td>
<td>16</td>
<td>8.8</td>
</tr>
<tr>
<td>FORTRAN</td>
<td>16</td>
<td>5.7</td>
</tr>
<tr>
<td>ALC</td>
<td>2.8</td>
<td>1</td>
</tr>
</tbody>
</table>

CHARACTERISTICS OF SAS USAGE

The vast majority of software at NCTCOG responds to ad hoc user needs. This is due primarily to the environment of a planning process in which data processing must continually answer "what if" questions. In order to respond to these data processing needs, both generalized and ad hoc software is available. Generalized software consists primarily of parameter-driven programs whose parameters can be changed quickly to respond to changing user requests. Where generalized software is not available, ad hoc programs must be written to specifically address the question at hand. As shown in Figure 4, a majority of data processing usage currently employs generalized software. In the past, the development of ad hoc software, currently representing 25 percent of the data processing requests, has accounted for a very large part of available programmer time. As can be seen from the figure, SAS has moved primarily into the area of developing this ad hoc user software. SAS has greatly increased the demand for ad hoc programs, as implied by Figure 5, but since much of the development of these programs is by the end user, the size of the programming staff has remained the same. SAS has not been used in the development of generalized software since there is no capability to use parameters supplied at run time to SAS programs.
The increased use of SAS to explore the available data base and the increased ease of answering "what if" questions makes it possible to have a better understanding of the data base on which the current transportation planning process is founded. This, hopefully, can lead to greater confidence in the planning process by both the planners at NCTCOG and the local governments who utilize the planning process.

SIZE OF PROGRAMMING STAFF

As mentioned before, the increased demand for development of ad hoc software with SAS has not required additional programmers. However, since increased programming is being done, it is useful to determine the "effective" size of the programming staff. There has always been a certain amount of program development that has been accomplished by nonprogrammers. This is estimated to represent approximately 10 percent of a programmer's time. However, the increased use of SAS has a much greater impact. Approximately half of the 300 SAS jobs run per month require a development effort which, as stated previously, averages 30 minutes per job. To have a programmer accomplish this development activity would also require approximately 30 minutes of communications time per job with the person making the request. Then approximately 150 hours (150 jobs x 1 hour) would be required each month if a programmer were to satisfy all current requests. This translates into an additional full time programmer position, as shown in Figure 6.

Furthermore, if this same number of requests were to be satisfied without the use of SAS, much more programmer time would be required. Figure 2 indicates that development in FORTRAN could require as much as ten times the effort, or ten programmers.

PROGRAM MAINTENANCE

Another impact of using SAS is that most ad hoc programs are no longer saved for possible later use, as was the case with previous FORTRAN programs. It is now easier to regenerate a program in SAS than to locate and modify a FORTRAN program. Previously, much time was spent in relearning what a particular program was doing so that it could be modified to meet current needs rather than concentrating on the concepts needed to solve the current problem. Several authors have estimated that maintaining and modifying software can account for up to 70 percent of the total cost of the software. It has also been suggested that even in languages such as COBOL and FORTRAN it may be cheaper to throw away old software and rebuild from scratch to satisfy required modifications. Therefore, SAS helps reduce the attempt to fit a previously square program into a round hole.

Some additional characteristics of SAS usage are shown in Figures 7 and 8. Figure 7 shows that SAS represents an increasing portion of the data processing dollar at NCTCOG. It seems that this will level off in the range of 20-25 percent of total processing cost. This assumption is made since most SAS users have reached the upper end of the SAS learning curve and, therefore, would be expected to continue the current type and level of applications.
CONCLUSION

Programmer productivity has been described as the potential restraint to the future growth of data processing. As has been shown here, SAS provides for much greater programmer productivity due to the high level of the language. In addition, since the procedures in SAS relate more to concepts in an application area or the results that are desired than to required procedural details in a language such as FORTRAN, an end user is more willing to use the language and can more quickly implement useful applications.

Continued increases in the processing efficiencies for input and output and the ability to use run-time parameters in SAS programs should greatly increase the use of SAS programs as generalized software.

REFERENCES