ABSTRACT

When an institution relies very heavily on SAS for its reporting requirements, the question arises as to the efficiency of SAS compared to a high level programming language. It is obvious that the ease of generating output with SAS utilizing its formatting, merging, and sorting capabilities is very economical in the short run. SAS is also economical when output will require many periodic format changes. However, if the programming requirements are very complex, require much data manipulation, and will be used over a long period of time, a program written in a high level language might be more efficient.

A client monitoring system for a Community Mental Health Center which was being written in PL/1 had to be re-written using SAS due to staffing changes and tight deadlines. This situation provided an opportunity to contrast a system written in SAS with the same system developed in PL/1. Development and run time costs are used to determine the point at which it becomes more efficient to write programs in a high level language. Also SAS's limitations are examined with respect to programming logic complexity and the use of DD's and ARRAY's is illustrated.

INTRODUCTION

The prominence of computer technology in the field of community mental health has become pronounced as the demand for accountability has increased at the federal, state, and local levels. In order to satisfy the reporting requirements, data must be collected for all phases of client treatment. Data collection and synthesis are somewhat complex in a community mental health setting in that many different forms of treatment are being administered in many different situations. For example some clients are inpatients in a mental hospital and are being seen by a staff member from the community mental health center, other clients are strictly outpatients, and some are participants in special groups such as mental retardation clinics. Community service and education is also provided by a community mental health center. Therefore it is difficult to design one data collection form which applies to all types of treatment and service. In addition to form design problems, output design is also a mammoth task. Many different and unrelated questions need to be answered. Information aggregation becomes essential. One solution could be to build a data base of all information collected on clients and then query this data base for each piece of required data. This can prove to be very inefficient in that data aggregation is just as important as data retrieval.

Very rudimentary data collection began at Valley Comprehensive Community Mental Health Center in March 1975. This initial development was reviewed and summarized by Deitz and Choban (1977). The output mainly consisted of tables which provided treatment information about clients and special groups and community consultation and education activities. This was accomplished using SAS's sorting, merging, and report generation capabilities. SAS was used exclusively for 4 years for all Valley's reporting. This decision was based on the fact that there was very little money available for computer personnel or computer time. Also since the staff had very little or no understanding of computer technology, it was difficult for them to specify what computer outputs would be useful to them. Since SAS can generate reports with very little programming effort it was decided to try to shape the staff's ability to explicate their needs by generating several different kinds of output with different variables and let them determine which would be useful. As the administration
became familiar with computer generated tables and as they learned how to use the information to make decisions about treatment programs, their requests for information became more complex. Also, they were better able to define what output should contain to be most beneficial.

From this developed the requirements for a very complex client monitoring system. This system of output combined and integrated much of the previously generated output and also answered many additional questions relating to client treatment. Valley by this time had hired a full-time computer programmer. Since the management thought they could explicitly define their requirements for the client monitoring system it was decided to write this system using PL/1 instead of SAS as it would be cheaper to execute. Also because of the logic complexity was doubtful that SAS would be able to satisfy the requirements. This system was to generate three different types of reports: 1) An Active Caseload Report(PART A), 2) An Inactive Caseload Report (PART B) and 3) A summary of parts A and B with the information categorized by treatment groups(PART C).

After PART A was about 60 percent complete, the full-time programmer left Valley to take another job. Deadlines were still tight and the new programmer needed time to learn about the center’s operation as well as finish the project. So, a computer consultant was given the task of completing the project. Initially it was still believed that PL/1 was the best development tool. The computer consultant spent 20 man hours trying to complete the PL/1 program but it became obvious that it would never be completed on time. SAS was then examined to see if possibly PART A could be generated for the first month to bide time until the PL/1 program could be completed. The result was that not only was the first month’s report generated on time with SAS, but also the whole system was completed in SAS. There proved to be no requirement that SAS could not fulfill.

SYSTEM REQUIREMENTS

The complexity of the system lay in having to manipulate two sequential files in many different ways. One file contained client termination information and could have duplicate entries if a client had been terminated, readmitted and terminated again. The second file contained treatment information. There was one entry for each service contact a client received. Furthermore each contact was classified into one of 14 service areas. PART A was generated for each staff member. It consisted of a list of all his/her “active” clients (those seen at least once in the last 90 days). Each client was listed once with his/her entry consisting of a summary of the service he/she had received. For each service category was listed the number of contacts, number of hours of service, date of last contact, and staff member delivering last service to the client could be seen by more than one staff member during the 90 day period. Also printed was first service and last service information and the number of days since client was last seen. The ordering of the clients was descending by number of days since last seen. If a client had not been seen for 60 days or more an asterisk was placed beside his/her case number to alert the staff member to check the client’s treatment status. A client could appear on different staff members reports since he/she could receive multiple treatments. Therefore the "staff member last seen" was used to trace a client to determine why he/she had not been seen in over 60 days. SAS’s ARRAY’s, DO’s and RETAIN statements proved vital in the generation of this report.

PART A listed non-terminated clients who had not been seen for 90 days but had been seen at least once during the three months preceding the 90 days. Also listed for each staff member was client case number, service last received, staff member delivering last service, date last contacted, and number of days since last contact.

PART C was a summary of parts A and B. That is, staff members were categorized into service areas and summary statistics were printed for each area. The report contained information such as number of clients receiving at least one service contact during the last 90 day period, service area, average number of days since last contact, number of contacts and number of hours of service for each area, clients receiving more service contacts than 90 percent of the active cases in each service area, etc. A summary of the inactive cases was also printed for each service area.

MACRO’s were essential to the organization of this output. They defined service areas, last day of the month of the current output generation, and service categories. In addition each part of the output was macro. PART A could then be printed for a certain month without printing PART B even though PART A had to be computed in order to obtain the PART C information.
COMPARISON OF PL/1 WITH SAS

PART A was 90 percent completed in PL/1. The sequential files were reorganized as an indexed sequential file with case number as the key. The system took 1.5 man months to develop and test and the program was over 600 statements long. The same section in SAS took 30 man hours to develop and test and was 260 SAS statements long. Run time costs were insignificant compared to development costs. Parts B and C took an additional 30 man hours to develop and test using SAS.

CONCLUSIONS

Even though a general purpose language like PL/1 is more efficient to execute than SAS, in some situations SAS is more cost effective. Research evaluation and accountability is a relatively new area, therefore reporting requirements have not stabilized. If the above program had been developed in PL/1 it would have had to run a fairly long time to recover its vast development costs and long running programs cannot be predicted at this time in Valley's situation. Also budgeting constraints in a community mental health center's environment do not allow for extensive programming effort. Currently SAS has proved to be the best software development tool given the limitations at Valley Comprehensive Community Health Center.

REFERENCES

PART A SAS STATEMENTS

MACRO CODEF
CODE=10:
IF $MCD<4 & ($MEN=1 | $WHEN=2) & $LOC<5 & ($REC<9 | $REC=14)
& $REC<15 & $REC=16 & $SER=0
THEN GJFT=11 #PHONE WITH COLLATERAL;

ETC...;

MACRO PARTA
DATA ALX:
FILE DEF:
INPUT $1 DATE MMDDYY6.
CCOL 1 2 COLZ 2
CCOL 6 6
TIME 25-31
PLACE 17 LOC 20 WHEN 32 MODE 35 REC 48-49
ACT 13-14
$7 MO $CHAR2
BY $ 9-10 YR $ 11-12
$92 CASE $CHAR6. SER 33-34 NAME $ 67
-20 $1 STAFF ID $CHAR4.
IF TIME=. THEN TIME=0;
TIME=TIME/86400;
CASECCR
STAFFCR
DATA ALL;
SET ALX;
CLODATE;
IF MC<MONTH(DX)-2 OR MC>MONTH(DX) THEN DELETE;
IF NAME=' ' THEN NAME=STAFF ID;
IF $SEEH='C000000 AND CASE='900000) AND ACT=1 AND SER=0 AN
D MODE=5:
CODEF
PROC SORT BY CASE CODE;
PROC MEANS NOPRINT BY CASE CODE;
VAR TIME;
OUTPUT OUT=R1 N=N SUM=SUM;
PROC SORT DATA=ALL;BY CASE CODE DESCENDING DATE;
DATA R11;
SET ALL;
BY CASE CODE DESCENDING DATE;
IF FIRST CODE;
KEEP CASE NAME DATE CODE MC BY YR;
PROC SORT DATA=ALL;BY CASE CODE;
DATA R11;
SET ALX;
BY CASE CODE;
RETAIN N=1-14 * NUM1-14 C1-C14 D1-D14;
ARRAY NUM (ij) NUM1-14;
ARRAY R (ij) C1-C14;
ARRAY W (ij) NUM1-14;
ARRAY D (ij) D1-D14;
IF FIRST CASE THEN DO;
CC=1 TO 14;
W=1;
C=1;
ENDIF;
END;
IF NAME=' ' THEN NAME=STAFF ID;
IF SER=0 AND (CASE='000000' AND CASE='000000') AND ACT=1 AND SER=0 AN
D MODE=5:
PROC SORT BY CASE;
PROC MEANS NOPRINT;
VAR TIME;
OUTPUT OUT=R3 N=N SUM=SUM;
PROC SORT DATA=R2;BY CASE YR MC YR;
IF TIME=0;
PERIOD=SUBSTR(NAME,2,12);
IF PERIOD='.' THEN NAME=SUBSTR(NAME,3,6); ELSE NAME=
IF LAST CASE THEN DO;
TOTALS=SUM(NUM1-14);
TOTALN=SUM(NUM1-14);
OUTPUT END;
DATA R2;
SET ALX;
IF SER=0 AND (CASE='000000' AND CASE='000000'):
PROC SORT BY CASE;
PROC MEANS NOPRINT;
VAR TIME;
OUTPUT OUT=R3 N=N SUM=SUM;
PROC SORT DATA=R2;BY CASE YR MC YR;
OUTPUT END;
DATA R4;
SET ALL;
BY CASE DATE;
LENGTH CODE 5 & 2 FRSTCD 4 LASTCD 4 FRSTSTF 5 LASTSTF 5 LASTDT 5
FRSTD 5

RETURN LSTAFF LASTSTF LASTCD FRSTCD CODE FRSTSTF FRSTD;
KEEP LASTDT LSTAFF LASTSTF LASTCD FRSTCD CODE FRSTSTF FRSTD CASE DAYS CCC;

CODE=*
CODE=CODE;

125
<table>
<thead>
<tr>
<th>CLIENT CASE NUMBER</th>
<th>PHONE WITH CELL</th>
<th>FACE-FACE WITH CLIENT</th>
<th>PHONE TRANSM.</th>
<th>MED REVIEW</th>
<th># AT WESTON</th>
<th>HOME VISIT</th>
<th>OTHER OUT-REACH</th>
<th>EVALUATION</th>
<th>OUT-PT COUNSELING</th>
<th>OTHER OUT-PT UNCAL, ETC.</th>
<th>PART CARE</th>
<th>OTHER</th>
<th>TOTAL</th>
<th>BROKEN APPT.</th>
</tr>
</thead>
<tbody>
<tr>
<td>* 1861</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>* 11526</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>* 400070</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**CASEWORKERS MONTHLY REPORT**

REPORTING PERIOD: 9/1/79 - 11/30/79

DATE: 12/15/79

CASEWORKER:  

ACTIVE CASELOAD INFORMATION

<table>
<thead>
<tr>
<th>1ST: NUMBER OF CONTACTS</th>
<th>2ND: NUMBER OF HOURS</th>
<th>3RD: DATE OF LAST CONTACT</th>
<th>4TH: STAFF DELIVERING LAST SERVICE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**PART A OUTPUT**

<table>
<thead>
<tr>
<th>FIRST SER-VICE</th>
<th>LAST SER-VICE</th>
<th>SS</th>
<th>SEE</th>
<th>REN</th>
<th>GEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>9/1 FAWLEY</td>
<td>9/1 TURNE</td>
<td>180</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9/1 FAWLEY</td>
<td>9/1 TURNE</td>
<td>180</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9/1 RAYBE</td>
<td>9/1 FAWLEY</td>
<td>178</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TOTAL**

<table>
<thead>
<tr>
<th></th>
<th>OL</th>
<th>AA</th>
<th>SY</th>
<th>ST</th>
</tr>
</thead>
<tbody>
<tr>
<td>9/1 FAWLEY</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9/1 TURNE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9/1 RAYBE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>