Abstract

This data management and forecasting system is a customized interactive financial system which supports data management for approximately 3000 construction projects funded by the Veterans Administration. Because none of the users are data processing personnel, the system is packaged and driven with menus and user-defined selection parameters coded as CMS EXECs. These EXECs communicate with the user in application-specific terminology. The EXECs translate user requests to runnable SAS code by writing SAS macros. In some cases EXECs are bypassed, and SAS programs conduct a user dialogue directly.

This system tracks actual project construction costs by month; it also provides detailed cost forecasts by month for the life of each project using base profile curves, adjusting for actual costs incurred. In addition, if a project's planned costs are known to deviate from the standard profile, its forecast may be adjusted to properly reflect any anomalies. Four customized reports and three plots are standard output. A variety of data base maintenance activities are also supported, as are ad hoc inquiries. This system also aids in budgeting and long-range planning by allowing the analysis of numerous alternative budget scenarios each year.

Introduction

The Veterans Administration's (VA) Office of Construction, Program Control and Analysis Staff, directed by Gerald A. Neumann, is responsible for preparing a construction budget for Congress and for the Office of Management and Budget each fiscal year.

A typical budget includes approximately 3000 construction projects which are funded by the VA. To assist in preparing this budget, the Office of Budget, part of the Office of Construction, identified the requirement for a flexible automated system which was dubbed the Construction Obligation and Outlay System (COOS).

The functions required of COOS are summarized below:

- What-if Analysis -- Includes on-going exercises showing numerous hypothetical funding situations each year.
- Tracking -- Tracking of actual costs by project by month for three different cost elements.
- Forecasting -- Projection of fifteen cost elements per project by month for 12 years based on history.

Standard Reports -- Required for Congress, for OMB and for internal decision-making.

- Ad Hoc Reporting -- To address specific information requirements.

Data Base Maintenance -- Ability to add, modify, or delete data stored in a central source as well as to archive data sets.

Selecting Software

SAS was chosen after having considered many other software alternatives, including INQUIRE, IDMS, EASYTRIEVE (products of Infodata, Cullinane, and Panaphic, respectively), and COBOL/VSAM. The combination of four features; system features made it difficult to find a good software fit. The software selected had to support the following: ability to easily and economically create "gaming" data bases holding only projects which meet certain selection criteria; ability of non-data processing users to produce their own reports after ad hoc retrievals from a data base; and the ability to easily generate and modify forecast data. The fourth requirement, having a working system very quickly, also influenced the selection of software.

Approximately 254 variables are associated with each project in the collection of data for COOS, which includes about 3000 projects. While a data management tool was necessary so that users would not have to count bytes to locate data values, a full-blown data base management system was really not required. The data base was not large enough to warrant a keyed structure such as INQUIRE or IDMS provides. Also, since smaller gaming data bases are created several times each week to support "what-if" analyses, it would have been expensive and time-consuming to create these as keyed data bases. None of the users have programming backgrounds, therefore COBOL or even EASYTRIEVE would likely have presented problems for users attempting ad hoc queries and reports.

Initially, SAS was not considered because of the paucity of statistics required after initial implementation. But, after examining many other alternatives, SAS was re-evaluated. It satisfied all four critical system needs, and provided many other benefits. For example, COOS' forecasting algorithms use three key construction dates to develop projections. Only SAS had flexible date handling and computation capabilities, which sped up the system development dramatically.

System Environment

COOS is usually run by budget analysts, not programmers. Periodically, management personnel who are unfamiliar with the system also must use it to produce "emergency" reports. For these reasons, COOS is a menu-driven system which uses
the vocabulary of the Construction Office. COOS is run under CMS, and the menu style of the system is implemented primarily using CMS EXECS. By using CMS EXECS, it was possible to develop generalized SAS software which is supplied with user-specified parameters at run time.

The development of COOS was segregated into two areas--SAS and CMS. Generally, CMS EXECS conduct the user dialogue, capture, validate, and store user responses, then invoke SAS programs. SAS programs then use the information stored by the EXEC in order to perform their function. Typically, several layers of EXECS are executed in COOS before a SAS program is actually invoked. This structure is described in more detail in the following paragraphs.

There are five central system functions in COOS; a main menu EXEC sets up all system libraries and terminal and printer attributes, then queries the user for which function is to be exercised. The response causes one of five other EXECS to be called or the user to be logged off. The hierarchical structure of the EXECS is shown in Table 1.

For example, upon logging on, the user is prompted with:

ENTER, REPORT, CREATE, MAINTAIN, AD HOC, STOP (1-6)?
(This prompt comes from COOS EXEC.)

A response of "2", for example, would cause REPORT EXEC to be invoked. REPORT EXEC then calls one of four specific report EXECS, or returns to COOS EXEC if STOP was requested.

Once in REPORT EXEC, the following prompt is issued:

BUDGET WORKSHEET, SPREAD REPORT, CONGRESSIONAL BUDGET, SPREAD GRAPH, STOP (1-5)?

Once the user selects a specific report, "1", for example, the EXEC for that report (BUDGWRST EXEC) is executed. There, the EXEC prompts for the parameters it needs to run the SAS program which produces the report. These parameters are then "passed" to the SAS program using macros. This passing is handled as follows:

1) EXEC prompts for parameters and stores these exec variables.

2) CMS file holding SAS macro names and '$' as contents is edited. The value for the parameter specified by the user in the EXEC and stored as EXEC variable is substituted for the '$'. The CMS file is saved under a new name.

3) SAS is invoked with the file just saved concatenated with the appropriate standard SAS program.

The following two dialogues demonstrate an abbreviated version of the prompting for parameters by EXECS which are then passed to SAS. Both dialogues are conducted by BUDGWRST EXEC.

Dialogue 1

MAJOR/MINOR/BOTH?
> MAJOR (User response)

Budget Worksheet is now produced for all major projects (where major or minor represents the funding level of the appropriation). Major projects are those projects (SAS observations) with 'MAJ' in the SAS variable APP. Minor projects have 'MIN' in APP. All projects are either major or minor.

Dialogue 2

MAJOR/MINOR/BOTH?
> BOTH (User response)

In this case, the Budget Worksheet is produced for all projects, major and minor.

An excerpt of BUDGWRST EXEC which calls SAS report program BUDGWRST SAS is shown in Table 2. The macro "BUDWAPP" is supplied to the SAS program BUDGWRST and is used by it. This macro selects all major projects, all minor projects, or is made a comment so that the appropriate projects are selected for the report.

The selection criteria for standard reports is quite lengthy; most report EXECS query the user for six or seven report parameters. The EXECS handle all validation of parameters thereby ensuring that when the SAS report programs are called, all macros which are passed are accurate and in the correct format. Similar styles of prompting are used in other system functions; for example, the input program asks which SAS data set (current fiscal year or any of the several gaming versions) should be used. The SAS input program is then passed this information as a macro.

The five main functions in COOS together invoke 21 specific functions. Each of these specific functions is composed of at least one EXEC which drives the SAS program or programs which provide the function, whether it is graphing a project's cost or backing a data base to tape. In a few cases, the EXEC conducts only a part of the dialogue and SAS conducts the rest. In these cases, SAS writes SAS code using results of the dialogue. This occurs most often when data in the SAS dataset must be consulted to set or validate a
parameter. Using SAS to conduct dialogue was kept to a minimum; the CMS style was easier to implement and maintain.

System Design

COOS' primary purpose is to supply budget data for OMB and Congress. Forecasts are done by month for the remainder of the current fiscal year and for eleven additional years. All monthly values must be retained, for once a project has entered COOS and been forecast according to the standard base profile, its own profile becomes the basis for new projections. One system function allows the user to adjust a project's monthly forecast to reflect any anomalies, therefore, each project's 144-month profile must be stored. However, annualized totals for the eleven forecast years are normally used for standard reporting and ad hoc queries, which account for much of the system's use. Therefore, the COOS data for each scenario (result of "what-if") is held in two datasets -- one holds all data needed for most typical system uses; the other data set is merged with the first only when forecasting or what-if analyses are done. The first dataset holds the annual totals by project for each cost element for each fiscal year as well as all descriptive data, including project number; the second contains only the 144 monthly values by project for each cost element and project number. Project number is the key by which the datasets are merged. This design minimizes operating costs, for most user requests can be satisfied without merging. The data set holding the 144 forecast values for each project are accessed only when the values are required. All datasets are in SAS "data base" -BUDGET.

System Development

SAS 7.94 and CMS Release 6 were used to develop COOS. Because the "top level" of the system (menus and dialogues) was implemented using EXECS, it was very easy and quick to develop. This division of labor also has simplified maintenance; for example, if an EXEC parameter's edit criteria change, no SAS code is affected. Only the appropriate EXEC must be modified.

The real computational and data management work in the system, however, is handled in SAS. The most difficult part of the system was the implementation of the forecast algorithm. The Office of Construction had developed rules for forecasting five cost elements for each of obligations, expenditures, and outlays. The five components which are forecast are the site amount, architecture and engineering amount, and completion, contingency and construction costs. The dates for the site, architecture and engineering, construction, and completion are input. Total cost of the project is input. Architectural and engineering percentages and contingency percentages are input for each project. To demonstrate the complexity of these rules, the rule for forecasting expenditures using input data and S-curves developed from historical data is summarized below.

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Site amount</td>
<td>Site amount is input</td>
</tr>
<tr>
<td>Architecture/Engineering (AE)</td>
<td>AE percent * (Totcost-Site)</td>
</tr>
<tr>
<td>Completion</td>
<td>.03 * (Totcost-Site - AE)</td>
</tr>
<tr>
<td>Contingency</td>
<td>Cont percent * (Totcost-Site - AE)</td>
</tr>
<tr>
<td>Construction</td>
<td>Totcost - Site - AE - Contingency</td>
</tr>
</tbody>
</table>

Timing:

Site amount is expended on site date. 95% of AE expenses is expended evenly over a period beginning on the AE date and ending on the AE end date. 5% of AE is expended on completion date.

Completion amount is expended evenly over a period beginning on the completion date and ending on completion date plus 18 months.

Contingency and construction costs are expended according to an S-curve over a period beginning on the construction date plus 4 months and ending on the completion date.

Projects are flagged each time actual data are entered, or when any of the input variables change. Then, when the forecast program is run, it recalculates the forecast only for those projects which have changed. All of this processing is handled in SAS except for one call to a FORTRAN subroutine which applies one of a family of predetermined S-curves to the amount and number of months to be forecast, using a polynomial interpolation to a tabled curve. The forecasting program, which also adjusts for any user changes to a project's profile, is about 2000 lines of SAS code. The costs for actually doing the forecasts are reasonable; however, they amount to only 20% of the total cost of running the program. A full 80% of the cost is incurred in compilation. Therefore, the capability to store a compiled SAS load module would reduce costs.
Specific System Functions

All system functions are provided using EXECs, SAS macros and standard SAS programs. Some of the more interesting examples of this packaging are described below.

1. **WHAT-IF ANALYSIS**

COOS allows the user to create as many gaming data bases as necessary to represent alternative budget scenarios. If the user wishes to create a new SAS data set called GAME1 from his fiscal year 1983 data base (FY1983) containing only projects with a TOTCOST greater than $1000 and AEDATE prior to January 1, 1984, the following dialogue takes place:

```
Prompt: DATA BASE NAME OF GAME
GAME1 (User response)

Prompt: USE WHAT DATA BASE
FY1983 (User response)

Prompt: ENTER 'STRTGAME;' TO
START, THEN SELECTION
CRITERIA, THEN 'ENDGAME;' THEN '/*'

(EXEC puts user in SAS)
```

(The following are commands entered by user)

```
11> STRTGAME;
21> IF TOTCOST GE 1000 AND AEDATE
   LT '01JANB4'D;
31> ENDGAME;
41> /*
```

SAS is called twice by the GAME EXEC. The first call, issued with NOERASE as an option, establishes the macros holding the data base names, passwords, and the STRTGAME and ENDGAME macros described above. The second call, also issued with NOERASE and NOWORKINIT, puts the user in SAS and gives him terminal control as well as access to the STRTGAME and ENDGAME macros. STRTGAME and ENDGAME are SAS macros composed by the EXEC. These macros are given below:

**STRTGAME** after macro substitution of data base names is:

```
DATA BUDGET.GAME1 (DROP =
data elements not required
in this dataset
PROTECT = PROTECT)
BUDGET.MGAME1 (KEEP =
data elements required
PROTECT = PROTECT);
MERGE BUDGET.FY1983
BUDGET.MFY1983;
BY PROJNO; %
```

**ENDGAME** is:

```
RUN; %
```

This way of allowing the user to create gaming data bases has worked very well. The number of data elements kept and dropped is over 100, so it would be tedious and error-prone to enter them at the terminal. The EXEC validates data base names for both data bases to ensure that the GAME data set does not exist and the data base used does then supplies the names to the macros. The user typically enters only one SAS command, an 'IF' statement, which is English-like and follows a prescribed syntax.

2. **CHANGE**

Changes to variables and deletions of projects are all handled by the user using PROC EDIT. The editing menu EXEC calls SAS twice. First the EXEC queries the user for the dataset to be edited. It stores this name by editing the SAS program PROCEDIT. In the example below, FY1983 was specified by the user. The EXEC then invokes SAS to run PROCEDIT SAS with NOERASE, which establishes the necessary macros.

**Excerpt of PROCEDIT SAS:**

```
MACRO EDIT PROC EDITOR DATA =
   BUDGET.FY1983
   (PROTECT=PROTECT);
   INFORMAT AEDATE ... DATE7; RUN; %
MACRO REP REPLACE CHGFLAG = 1
   DESCFLAG = 1%
```

The EXEC then invokes SAS again with NOERASE and NOWORKINIT. At this time, the user types EDIT; and is placed in
PROC EDITOR. The macro "REP" does not include a semi-colon, so the user types REP then his changes to variables. The macro REP turns two flags on (transparent to the user) and makes the changes that the user specified. The flipping of the flags has been very useful in COOS. Since PROC EDITOR cannot do user-specified error checking, a validation program is run to do this editing. This program does "table look-up" using format libraries, does range checks on numeric variables, dates, etc. The program only checks records that have been changed since it was last run (i.e., DESCFLAG is 1). The validation program resets DESCFLAG to 0 after checking all required variables and printing all error messages.

The CHGFLAG variable is used by the forecasting program. It is set in both the program where actual expenditures, etc. are entered, and in PROC EDITOR, since if either of these programs alters a project, it must be reforecast. When the forecast program runs, it only reforecasts projects which have CHGFLAG turned on. After recalculating a project's forecast, the program sets CHGFLAG back to 0 for that project (observation). These flags have reduced operating costs, for most often only 10% of the projects need to be reforecast or validated.

Production Environment

COOS went into production in September of 1981. SAS' features have enhanced its daily operations as well as the system development. For example, in order to perform an ad hoc query of the data base, the user is put into SAS by the controlling EXEC. Then, the user typically enters a subsetting IF and runs a PROC PRINT. The ad hoc query, as entered at the terminal, is captured and stored as ADHOC SASLOG (a CMS file). Therefore, if a query did not run properly, and a SAS consultant is not available on-site, the query may be viewed by an off-site consultant exactly as entered simply by typing the CMS file ADHOC SASLOG.

The SASLOG files have proved to be very helpful to technical consultants. Most often, if a program does not run as expected, the reason can be found in the SASLOG.

Also, all user-defined parameters are stored both in the SASLOG and in the current run parameter file. Again, if a user experiences difficulty with the system, a SAS consultant may go back and see exactly what parameters the user entered.

Finally, the flexibility of SAS is a great asset for COOS. As the Office of Construction has become more familiar with new COOS and the features of SAS, they have been able to create more sophisticated queries and accompanying reports, to create new gaming data bases by merging smaller ones, to move certain projects to a temporary data base, modify forecasts and, if needed, copy the new forecasts back to the main data base. The users also will be able to take advantage of the statistical capabilities of SAS when they have more historical data accumulated. At that time, the base S-curves currently used in forecasting may be fine-tuned to reflect the additional history.

In conclusion, SAS and CMS together create an excellent environment for COOS. It is believed that using SAS helped to reduce both development time and costs substantially, and offers significant potential for expansion of the system to meet user needs using endless other SAS features.
TABLE 1

TABLE 2

Excerpt of BUDGWRST EXEC

- APPTYPE
  &UTYPE MAJOR/MINOR/BOOTH
  &REAS VARS &APP
  &IF .&APP EQ. &GOTO -APPTYPE
  &IF &APP EQ STOP &GOTO -END
  &IF &APP NE MAJOR &IF &APP NE MINOR &IF &APP NE BOTH &GOTO -ERRAPP
  &APP = &SUBSTR &APP 1 3
  &GOTO -SELECT
- ERRAPP
  &UTYPE REENTER
  &GOTO -APPTYPE
- SELECT
  STATEW MACBUDW SAS (NOTYPE
  &IF &RETCODE EQ 0 ERASE MACBUDW SAS
  &STACK VERIFY OFF
  &STACK TOP
  &STACK NEXT
  &IF &APP EQ BOT &STACK C/ IF/ */
  &IF &APP NE BOT &STACK C / $/ &APP
  &STACK FILE MACBUDW SAS
  EDIT MACBUDWS SAS
  SAS MACBUDW BUDGWRST (NAME BUDWRST)

MACBUDDS SAS (This file is edited and filed as MACBUDDS SAS.)

MACRO _BUDWAPP IF APP = '$' ;%

MACBUDW SAS (After Dialogue 1)

MACRO _BUDWAPP IF APP = 'MAJ' ;%

MACBUDW SAS (After Dialogue 2)

MACRO _BUDWAPP * APP = '$' ;%