THE SAS SYSTEM CAN REPLACE TRADITIONAL PROGRAMMING LANGUAGES
Judith H. Mopesk, ORI, Inc.

1. INTRODUCTION

A 50,000 line production system, developed by ORI, Inc. for the Bureau of Labor Statistics (BLS) was entirely programmed using SAS software. The system is modular in concept, easy to modify and maintain, and most of the programs were coded, tested, integrated and put into production in less than twelve months. Generalized production-level testing and verification programs were also written using SAS software. This system is an important component used in the construction of the Consumer Price Index. This paper will summarize the requirements of this application and then demonstrate how the SAS programming language can be used to implement production systems.

ORI developed a Data Adjustment System for the Consumer Expenditure Interview Survey. This survey collects data concerning a huge variety of expenditures on a continuing basis and is used to update the "Cost Weights" required for calculation of the Consumer Price Index. The system includes capabilities to perform complex computations (to combine data or transform data into new fields), imputations (to replace missing or invalid data for critical attributes, and expenditures), and allocation (to distribute expenditures reported in combination to individual line items).

The major requirement for the Data Adjustment System was that all programming be done using the base SAS product. The requirements for documentation and maintainability were those that are needed for all large systems. A completely documented, production system had to be turned over to BLS by September, 1983. The computer programs had to be maintainable by staff other than the originators.

The technical requirements dealt with the programming of the methodologies needed to calculate new variables, to impute for missing or invalid data, and allocate combined expenditures. The techniques for imputing values fell into three main categories: Hot Decking, Weighted Means, and Percentage Distribution. When expenditures were reported as combined items (e.g., hospital stay, utilities, home maintenance), the cost had to be allocated to its component parts. The allocation was based on values reported for each separate item. Several allocation methods were required, depending on the expenditure reported in combination.

Output processing and reporting requirements were common to all of the 40 expenditure sections. Error reports were required whenever conditions fell outside of the prescribed rules for data adjustment. Intermediate results were reported for all methods. This included weighted means and weighted counts prior to and after data adjustment. Expenditures that could not be automatically imputed/allocated due to insufficient valid source data were identified for manual adjustment. The manual adjustment reports had to contain sufficient detail on the household's other expenses so that a value could be hand-calculated. All successfully adjusted attributes and expenditures were written in the format required to update the database.

The above requirements were given to ORI in September, 1981. A small staff of analysts, working part time, embarked on the project. Analysis techniques defined by Tom DeMarco[1] were used during the first phase of the project. A SAS-based MACRO driven Data Adjustment System was designed and successfully implemented in September, 1983 which satisfied all of the above-mentioned requirements.

2. LIFE CYCLE OF A SYSTEM

The SAS System is very easy to integrate into the life cycle of a large programming project. A project life cycle can be separated into eight phases:

- Analysis of End-User Requirements
- System Specification
- System Design
- Program Design and Coding
- Prototype Development
- Unit Testing
- System Testing
- Final Documentation

2.1 ANALYSIS OF END-USER REQUIREMENTS

Upon receipt of end-user requirement documents for a project, the staff must set out to understand what the authors desired by gathering and organizing all related documentation and meeting frequently with the authors to clarify ambiguous points. Using sample data that are available, small SAS programs can be run and descriptive statistics examined to verify that key data fields are accurately described and coded. Reports should be made back to the end-users that detail problems, often with suggested alternative solutions and descriptions of the advantages and disadvantages of each alternative. These small programs can be incorporated into the production system as a way of verifying
results. The advantage to using the SAS System during the research and analysis phase is that small studies and detailed reports can be prepared in a timely manner.

In the early phase of the system's life cycle, major system components, such as data definition and algorithms, must be studied in great detail. From the beginning of a project the problem of staging the design and development of the major system-level components must be addressed. Which functions should be designed first? What is the inter-relationship between functions? From the start of a project, the staff must consider the programming language to be used for implementation.

2.2 SYSTEM SPECIFICATIONS

O'RI has found that the use of Structured Design and Analysis Methods, as defined by Yourdon and DeMarco, to develop functional specification documents from user requirements has numerous advantages. The functional specification provides a basis for the mutual understanding of the problem between the users and the designers. The detailed process assumptions, descriptions, data flows and data flow diagrams inherent in this methodology, enable the programmer to transform the specification into a program or system design. The process descriptions have almost a one-to-one relationship to SAS Data Steps and PROC Steps. The relational, or flat data structures of the SAS Data Sets are analogous to data flows.

2.3 SYSTEM DESIGN

The major problem in converting a system specification into a system design is translating the non-procedural format of a functional specification into the details of a traditional design for a procedural language. Using the SAS System, which is simultaneously procedural (in the DATA Step) and non-procedural (in the PROC steps), the transition from the functional specification to the design phase is simplified. Given the non-procedural structure of SAS code and the fact that SAS programs are a sequence of DATA and PROC steps interconnected by common datasets, there is a natural one-to-one relationship between data flow diagrams and the SAS implementation of the design. A process is designed as either a DATA or PROC step. A dataflow connecting two such processes is the creation of a SAS data set in one and the reading of it in another. In fact, the use of the SAS programming language maximizes the modularity of programs and systems since the dataflows connecting various processes are physical entities themselves (SAS data sets) rather than just logical representations. The connection between processes is an entire data set passed all at once, rather than single observations passed one at a time. The modularity between processes leads to more efficient development, testing, debugging and maintenance. It is true that this one-to-one relationship between data flow techniques and SAS could be mimicked by other languages; however, there is no need to do this given that SAS programs can be efficiently designed, quickly coded and easily implemented. Figure I shows a data flow diagram and its translation to a SAS program design.

Once similar system level functions are identified, the staff can explore the various SAS programming techniques that can be used to perform these processes. In a long-term project where a large amount of code has to be developed, generalized code to perform common processes (e.g., report writing) is desired. The use of subroutines becomes the design choice, and the SAS System satisfies this design requirement through the powers of the SAS MACRO facility. Therefore, the SAS System becomes a viable alternative to traditional programming languages.

MACRO execution in the SAS System performs analogously to subroutine execution, and the interface between a main program and a MACRO is via a SAS Data Set(s) and "parameters" passed as MACROs or MACRO variables. Through the use of generalized code which is implemented as system level MACROs, application program designs are constructed so that each application in the system only has to deal with its unique functionality. The interface with the system level MACROs becomes as simple as a subroutine "CALL". Furthermore, if later changes and enhancements to system level applications are required, the application programs are not affected. Only one set of code needs to be changed—the MACRO. This is particularly useful in incrementally developed systems.

Another advantage of MACRO driven systems is that functions implemented as MACROs are tested by many applications. Since the same set of code is used over and over again, the code itself becomes reliable.

The problem of conditionally executing Data Steps and PROC Steps in the SAS System is solved by using the MACRO facility and the %INCLUDE statement. Conditional execution limits the commitment of computer resources to actual, rather than potential needs. A result of the use of the %INCLUDE and the SAS MACRO language capabilities, is a computer system design that very closely parallels the Functional Specification documents.
DATA_ELEMENTS
SELECT
ELIGIBLE
RECORDS

PRODUCE
REPORT

CALCULATE
NEW
VARIABLES

ERRORS

ERROR-FILE

UPDATES

UPDATE-FILE

DATA

DATA_STEP
READ ALL DATA
KEEP ELIGIBLE RECS

DATA_STEP
CALCULATE NEW VARS.
SUM BY GROUPS...

PROC SORT
BY N1 N2

PROC PRINT
BY ;
ID
LABEL;
FORMAT;

REPORT

PROC PRINT
ID ;
SUM ;

REPORT

UPDATE-FILE

* SAS DATA SETS
** OS FILE

FIGURE 1 DATAFLOW DIAGRAM AND SAS PROGRAM DESIGN
A modification request for one of the allocation methods in the Consumer Expenditure Survey provides an example of how a MACRO driven design permits quick changes in an existing system. This request was not a functional change, but a detailed algorithmic change. The change was easily made by invoking PROC UNIVARIATE (to compute percentiles) instead of PROC MEANS (to compute means). Since all allocation methods in this system were implemented as MACROS, the change was easily made with no impact on the applications programs.

2.4 PROGRAM DESIGN AND CODING

After the functional specifications and application program designs are completed, the program functions must be decomposed into SAS Steps. Flow charts can be used to identify the SAS data sets needed to link the Data Steps and PROC Steps. Design walkthroughs held after this process, but before coding begins, provide the staff programmers and the designers with an opportunity to learn about the program and address details that might have been overlooked. It is important to mention, also, that the author of a functional specification does not necessarily have to be the programmer. Consequently, early walkthroughs serve as a means of clarifying a programmer’s design to an analyst.

Once the program design is completed, coding commences. Due to the explicit relationship between flow charts and SAS steps, programming responsibilities can be turned over to junior level SAS programmers, under the guidance of an experienced SAS programmer. SAS programs are implemented incrementally, i.e., each data step is thoroughly tested before additional code is added. Frequent lower-level testing makes the task of debugging hundreds of lines of code much easier. Liberal use of PROC FREQ and PROC PRINT provides easy validation for testing logical program conditions.

Program coding conventions must be adopted for a system that will be implemented by several different people. A manageable set of coding standards must be agreed upon. ORI has established such a set of SAS coding conventions. One category of conventions, for example, sets the standards for program documentation. Detailed comment sections are required at the front of each Data Step, and “DO-END” loops are always commented. PROC PRINTS are executed after each data step to verify that correct data transformations occur. While these “test prints” should be delivered for user review, they are not desirable as part of production runs. However, “test prints” can not be eliminated from the programs because the code may have to be changed later. These prints can be “commented out” with the use of the asterisk “*” so that if future changes are requested, the “test prints” are already in place. The MACRO facility can also be used to accomplish this (Figure 2). A poster paper presented at this conference “Improving Programmer Productivity with Coding Conventions” details the SAS coding conventions established by ORI.

2.5 DEVELOPMENT OF A PROTOTYPE SYSTEM

In the development of any large-scale computer software system, it is important to present the ultimate users of the system with an early release (or version) of production output. The users work with this data to determine potential weaknesses in methodology or the need for different types of printed output. In the early months of a large software development project, these changes are easily implemented. Once several thousand lines of code have been tested, change requests can result in delays to the schedule.

This type of “prototype” development is also important to the designers of the application software. During prototype development, the feasibility of new ideas and approaches to problems can be tested. Benchmark runs can be made to test the efficiency of various techniques. Staff members are willing to “brainstorm” when there is no immediate pressure of deadlines.

Experienced SAS programmers can quickly code, test and implement a prototype system, which is a skeleton of the final product. One component of the system can be transformed into a “mini” system by linking a few SAS Data Steps and PROC Steps together. The prototype can be packaged as if it were the final system, using the expected JCL references and system command procedures. Variables are easily created in the SAS system because they are only referenced by name. PROC PRINT can be invoked wherever detailed reports are to be produced. Programming a prototype for a SAS system does not have to take a long time.

The development of a prototype provides for early end-user feedback and technical feasibility testing. The prototype developed for the Data Adjustment System demonstrates this. Four applications were designed, coded, tested and packaged as a system prototype.
Each application required different data adjustment methods which were packaged as MACROS. Once all four sections were completely tested independently, all MACROS were made a part of a MACRO library and all adjustments were set up to execute conditionally. A production run was made for the prototype which demonstrated ORI's ability to implement the application software in a total SAS environment.

To assist the end-users in review of the printed output, ORI staff held several "output" walkthroughs. Detailed explanations of the printed output helped the users determine whether they needed more or less output, reformatted output, or different kinds of statistical measures of results.

The prototype served as the "standard" for all future application software development on this project. After each of the remaining system-level MACROS was developed and thoroughly tested, a code walkthrough was held and test results were presented to the users. Usage documentation was produced before each MACRO was released for general programmer use.

2.6 UNIT TESTING

Behind a successful computer application system is a comprehensive plan for testing the code. ORI has set very high standards for testing SAS code and reviewing printed output. There can be no replacement for the thorough testing of each SAS Data Step. The data sets created must be verified by hand and all logic paths must be tested. Peer review is essential.

To hand-verify each step, descriptive frequencies and prints of each data set are generated. Printouts of new data sets and data transformations are hand-checked against the original frequencies to verify that all of the newly created data sets contain the correct results (e.g., variables and observations).

Once data are hand-verified for all regular, expected conditions, test situations must be set up so that all logical paths through the program can be forced to occur. Rather than create test data sets, test case situations are easily generated in the SAS language by selecting or creating subsets of data in a "pre-processing" SAS Data Step. frequencies run on the selected cases (via PROC FREQ) identify the testing environment.

Programmers are also required to test for unusual cases to insure that the application programs do not terminate abnormally. All situations which force conditional execution must be demonstrated.

2.7 SYSTEM TESTING

The first step in system testing an application program is to package the program in the production environment. The second step in system testing is the presentation of acceptance tests for end-user review. Before the start of testing, analysts meet with the users to help them determine what kinds of data conditions they need in order to verify that their algorithms work correctly and accomplish the desired results.

FIGURE 2 EMBEDDING TEST PRINTS IN PRODUCTION CODE - TWO APPROACHES
2.8 FINAL DOCUMENTATION

Throughout the life cycle of a large system, documentation must be produced which coincides with the completion of specific milestones. The functional specification documents contain the detailed analysis of the user's requirements. The program design documents address the overall system considerations. The system level MACRO documentation contains detailed usage instructions. The SAS coding standards should be documented before any code is written. Once the system is tested, the documentation has to be finalized. It is important that the documents be updated to reflect design changes that occur as a result of walkthroughs, peer review and end-user acceptance.

An example of the documentation for the Data Adjustment System, excluding the code which resides both "on-line" and in an archival file, is presented below:

- Functional Specifications for Interview Data Adjustment.
- Functional Specifications for Data Adjustment Methodologies.
- Important Information For New Users (including SAS coding conventions)
- User Documentation for System Level MACROs
- General Purpose Programs, MACROs and Utilities

3. CONCLUSION

The use of the SAS System is moving away from its exclusive use for report writing, data reduction and ad hoc applications. This paper has demonstrated how the SAS language can be integrated into system development. This can be summarized as follows:

- ANALYSIS - Small SAS studies and detailed reports can be prepared in a timely manner.
- SYSTEM SPECIFICATION - The basic SAS structures are easily extracted from Structured Design and Analysis Methods.
- SYSTEM DESIGN - Conditional execution of SAS steps and the power of the SAS MACRO facility provide the SAS programmer with the tools of traditional, procedural languages.
- PROGRAM DESIGN AND CODING - Flow charts become SAS steps linked together with SAS Data Sets. SAS PROCs provide capabilities that are not quickly developed in other languages. Less experienced programmers can work effectively from SAS designs.
- PROTOTYPE DEVELOPMENT - The SAS system enables quick development of a "mini" system. This serves to demonstrate the feasibility of new ideas and identifies potential design weaknesses.
- UNIT TESTING - Test prints and subset selection are easily incorporated into SAS programs as conditionally executed code. Re-testing is easier since the verification tools are already debugged and in place.
- SYSTEM TESTING - SAS PROCs can aid the end-user in verifying results, and accepting the system.

The SAS system is a strong contender for addressing the requirements of large-scale data processing applications.

REFERENCES:


The author can be contacted at:

ORI, Inc.
Software Applications and Training Program Office
7910 Woodmont Ave., Suite 1405
Bethesda, MD. 20814
(301) 656-3276