CAN SAS REPLACE TRADITIONAL PROGRAMMING LANGUAGES IN LARGE SYSTEMS DEVELOPMENTS? - A MANAGER'S PERSPECTIVE

J. Harvey Trimble, Jr., U.S. Department of Labor

BACKGROUND

SAS has three principal features that make it more desirable than traditional programming languages for large system developments:

- Data independence;
- Nonprocedural facilities; and
- Relational or Flat data structures.

SAS's data independence which permits data to be defined once and used from then on by name only, is an extremely valuable asset for large system development. It insulates applications from one another which reduces the impact of requirement changes. It also permits the development of very flexible applications which are loose collections of data transformations.

Nonprocedural facilities are those which permit the user to express what needs to be done but not how to do it. SAS's nonprocedural facilities permit complex calculations and data transformations to be done in a very few high level language statements. These facilities reduce the coding and testing time required to develop an application. They also provide substantial application validation support in that it is quite simple to analyze the inputs and outputs to any transformation with PROC PRINT's and PROC FREQ's.

Relational or Flat data structures are simply two dimensional tables where the rows are records or observations and the columns are variables or data elements. These Relational or Flat data structures are easy to understand, and help limit application complexity since the data structures cannot in themselves be very complex.

THE MANAGER'S PERSPECTIVE

This paper is subtitled a Manager's Perspective because an individual's perspective changes on the career path from technician to manager. A manager's perspective differs from the technician's point of view in that a manager must be concerned with the broad issues of system development cost, reliability, flexibility, and operational cost rather than on narrow technical issues. The manager must assume an organizational view of the development while the technician is allowed to have an application level view.

These different perspectives are important to recognize since using SAS as a replacement for traditional programming languages is a fairly radical change in system development approach, and may therefore face resistance. This resistance has two dimensions. First, the technicians may simply resist the change for traditional reasons, e.g., anxiety, or concern for status. Second, the technicians may resist the idea because the use of a package like SAS may be perceived to be counter-productive to his/her professional development since they may not perceive SAS experience as being a marketable skill. These reasons for resistance can be dealt with by providing the staff with training in the use of these techniques and by following the principles outlined below.

THE APPROACH

We succeeded in replacing traditional programming languages with SAS in two systems which store, process, and tabulate Consumer Expenditure data from a quarterly survey of 4000 families and consist of 650,000 observations and 3000 variables. Our success is a result of following these principles:

- Develop strong management interest and involvement;
- Thoroughly plan the development;
- Separate analysis from design;
- Obtain expert advice;
- Develop measures of applicability; and
- Use SAS where it best fits.

DEVELOP STRONG MANAGEMENT INTEREST AND INVOLVEMENT

Strong management interest and involvement is critical to making a change over to SAS. Technical change is very hard to introduce unless all management levels agree on the mandate for change. Since any change is disruptive in the short term, there must be strong management commitment to the change or the short term argument that "we could do this faster/better in COBOL" will prevail. The Project Managers must take responsibility for design decisions and resource commitments in order for SAS use to become permanent.

THOROUGHLY PLAN THE DEVELOPMENT

Planning the development of a large project requires a substantial commitment of time and resources. This commitment will pay dividends throughout the life of the project, and must be sold to all management levels on this basis. It took us nine months to define the Consumer Expenditure Survey's 450 activities and develop an automated schedule which was constrained by our available resources and matched our management instincts on how the project should proceed.

By reporting our progress against this schedule, we showed the effects of each week's effort on the immediate and long range targets. This information permits managers to evaluate various resource utilization strategies, and take timely corrective action to keep the project on schedule. This kind of planning creates the
ideal management climate to demonstrate the effectiveness of replacing traditional program-
ing languages with SAS. It also provides the perspective necessary to support the heavy emphasis on analysis inherent in Structured Analysis techniques.

Separate analysis from design

In order to permit management review and control of the technical development, it is vital to develop a means for separating systems analysis from systems design. Analysis must define "what is to be done" and leave the "how it is to be done" to design. Separating the two permits the postulation and evaluation of different design approaches which lets the prudent designer pick the best.

More importantly however, doing rigorous analysis establishes an excellent framework to do the job right the first time. It demands that all concerned parties invest sufficient time and energy in defining what is to be done before any resources are committed to doing the job. This results in reducing the magnitude and volume of requirements changes at either the design or implementation stage.

Myers, Constantine, and Yourdon approach to Structured Analysis is an excellent method for doing Systems Analysis. Tom De Marco's book "Structured Analysis and System Specification" offers an outstanding guide to the why's and how's of structured analysis. This approach fits well with the internal design of SAS; the complexity of transforming the logical analysis into a physical design is reduced using SAS as the application language.

On the other hand, using Structured Analysis and Design techniques in connection with traditional programming languages creates a serious problem in the transition from analysis to design. The reason for the varying degree of complexity between the two approaches in making the analysis to design transition is because Structured Analysis techniques stress the functional, i.e., nonprocedural, aspects of problem definition. It is therefore much easier to design systems which use nonprocedural facilities (e.g., SAS) than it is to proceduralize the design for traditional programming languages. While De Marco's book leads you to believe that the transition from analysis to design is straightforward; it is not.

In fact, the questions of how many application programs need to be developed, and how the programs should be structured are not well addressed by this methodology. With SAS applications however, we developed a data step or used a procedure for almost every data transformation in our analysis. SAS's data independence provides substantial design flexibility since the design can be changed to suit new requirements with little impact on existing applications. Application developments using SAS yields less complex System Architectures than those designed using traditional programming languages. Application complexity is reduced because the simple data structure of SAS data sets forces each data step to deal with simple data transformations.

Also, only a very small amount of control logic is required to implement a SAS application, because many transformations can be done with SAS procedures. Even when data steps are used, control logic is reduced because most of the input and output logic is in the SAS Supervisor. Eliminating the development of control logic reduces both the time to write and to test the application because control logic is complex to write and hard to thoroughly test.

Obtain expert advice

Introducing a new way of doing business to an experienced staff requires a short term commitment to training and a long term commitment to having expert guidance available in how to apply the new techniques. We started by having the whole staff take a week long course in Structured Analysis and Design and a three day course in SAS. This familiarized the staff with the concepts, but in no way provided them with enough experience to proceed alone through the development.

Structured Analysis techniques are sufficiently fault-tolerant to allow the natural on-the-job training experiences to fill the gaps left by the classroom. By their very nature, Structured Analysis techniques work to capture the requirements from the top down, and while our early efforts described the requirements in overly physical terms there was a natural tendency to work back to logical presentations.

Using SAS as a system development language, on the other hand, requires fairly substantial adjustments in thinking for most computer systems professionals. How well a technician adjusts to using nonprocedural facilities and flat data structures is related to how much training the individual has had in functional analysis. Here again, the training in Structured Analysis pays off in preparing the staff for a new way of thinking.

However, we did find it necessary to obtain outside expert help to develop alternative designs and explain how SAS works internally. We contracted for outside SAS expertise to provide two basic types of services: (1) on-site consulting, design review, and code critique, and (2) specialized training in directly applying SAS to the typical statistical data processing application. These services proved to be quite essential to overcoming staff resistance and in developing efficient and effective applications. It was vital to have an outside expert develop and review design alternatives in the early stages of introducing SAS. You cannot expect a skilled technician to be comfortable with any approach besides what they know best.
DEVELOP MEASURES OF APPLICABILITY

Introducing this kind of a change to an experienced staff is aided by having measures to evaluate when to use it. These measures need to be publicly known and understood, and fairly applied. You must avoid being dogmatic about introducing the use of a new approach. Measuring the applicability of using SAS in any given application makes the selection of a design approach a defensible and rational process. This permits the technicians to differ professionally on how to do the job; avoids some defensiveness; and also helps avoid applying a tool where it is not well suited.

The measures we used for our applications are:

- Minimize design and application complexity;
- Minimize development costs even at the expense of operational costs;
- Maximize application flexibility and maintainability; and
- Use SAS except when it is demonstrated to be unsuitable.

The only one of these which needs elaboration is the measure to minimize development cost even at the expense of operational costs. This measure goes to the heart of the technical claim that a more efficient application could be developed if a traditional programming language were used rather than SAS.

This argument is a variant of the Assembly Language versus COBOL controversy, and is easily addressed in a management context if one looks at total life cycle costs when evaluating the payoff for higher operational performance. That is, even assuming a more efficient application could be developed in an alternative to SAS (a questionable assertion in many cases), the cost of developing that alternative must be used to evaluate the possible savings. Most claims of SAS inefficiency reflect design problems rather than poor performance of SAS. SAS is quite an efficient tool except in the conversion of large files into SAS data sets and vice versa.

USE SAS WHERE IT BEST FITS

One key to effective use of SAS as a replacement of traditional programming languages is to be selective about where to apply it. Early success is important to overcome the technical resistance to its use, and vital to show management that it has backed a winning strategy. Initial applications must be carefully evaluated to be sure that they are well suited for development in SAS.

Phased developments or pilot applications are especially desirable to pursue in these initial SAS applications in order to quickly reapply what is learned from the first experiences. Minimizing the length of time between analysis and installation encourages small developments, and shows visible progress toward the final goal.

The principal area which we have chosen not to use SAS for is data editing and data base updating. While SAS has some editing facilities we chose to develop our own generalized data base maintenance application in PL/I which used the host language interface to our data base manager.

We chose a relational data base manager, RAPID, developed by Statistics Canada to manage our data. We did this to keep the cost of small volume updates against data bases of between 50,000 and 500,000 records low; to permit the direct processing of the data by PL/I programs for those applications we choose not to do in SAS; and to permit low cost selective data retrieval. We developed a SAS user procedure to directly connect the data bases with SAS, which has worked exceptionally well.

The single most valuable aspect of this relational data base manager, however, is that its data structure matches SAS's data structure perfectly. SAS data sets are created from our user procedure which are physical subsets of the data base. The column names used for the relational data base manager are proper SAS names so variables are referenced by the same name in both systems. The data manager also stores its data definitions with the data base so both systems are data independent. The cost to retrieve data from our relational data base and convert it into the internal SAS format is completely offset by the leverage and flexibility offered by SAS.

CONCLUSIONS

In summary, SAS is an excellent replacement for traditional programming languages for system developments. Its nonprocedural capabilities and data independent features make it an exceptionally productive system to use. The driving forces in minimizing system and application complexity are to thoroughly analyze each application and keep data structures as simple as possible.

Structured Analysis and Design techniques coupled with the relational model, which SAS uses naturally, are ideally suited to maximize development success. They work well together for any size project, and offer the Project Manager substantial peace of mind that what is being developed is what the customer wants, and that the product will work well and will be highly maintainable.