INTRODUCTION

The U.S. Bureau of Labor Statistics Division of Consumer Prices and Consumption Studies (CPCS) develops, maintains, and runs the software that processes much of the survey data used in the Bureau's Consumer Price Index (CPI) and Consumer Expenditure (CE) programs. CPI and CE production activities involve multiple concurrent production cycles for different computerized operations that include sampling, survey form generation, survey data capture, data imputation, and index computation.

With the exception of PC-based data capture software, all the production applications at CPCS are written in SAS software and are run under MVS in batch mode using SUPERWYLBUR for file editing and job submission. Production is run mainly at night or on weekends, and each survey has a staff that maintains the production software and runs the jobs. Each survey also has a development staff that implements changes and enhancements to the applications. In order to minimize JCL errors and streamline the production cycles, CPCS developed a batch job interface in the mid-1980's that became known as the Production Control System, or PCS. The PCS prepares JCL streams based on user responses to on-screen prompts, and eliminates the need for users to edit JCL manually when running production jobs.

The PCS was written in the SUPERWYLBUR command language using a complex system of editing commands stored in macros. These macros prompt the user for information and substitute user responses for bracketed tokens in the job's JCL skeleton. Users can change parameters in existing JCL lines; add, comment, or delete lines of JCL and SAS code; and review or edit the JCL file before submission. Though the PCS does not have full-screen user interface capabilities, its strength lies in SUPERWYLBUR's powerful text editing features such as column-oriented text manipulation and command "piping". The PCS takes full advantage of this editing capability to do its JCL file tailoring.

Though the PCS had been running satisfactorily, its line-by-line prompting mode had become obsolete, and support for SUPERWYLBUR would be problematic when BLS contracted with a new computer center in 1992. For these reasons, CPCS decided in the spring of 1991 to move to a new environment for batch submission. Atlantic Research Corporation (ARC) was asked to do the design and lead the development of the new system. ARC's challenge was to develop new software that preserved and improved the PCS' functionality while adding a better user interface.

THE PCS/AF SYSTEM: AN OVERVIEW

Before discussing the development process which is the direct purpose of this paper, it is worthwhile to give at least an overview of the new system. This overview will serve to help illustrate both the power and complexity of the new system and the great differences in human interface and technology between the old and new systems. The name chosen for the new job submission system was "PCS/AF." A functional overview of the system is presented in Figure 1. Written in SAS Screen Control Language (SCL) under Version 6.06 of the SAS System for MVS, PCS/AF allows users to write, prepare, submit, monitor, and retrieve MVS JCL batch job streams from within a SAS Display Manager session. PCS/AF contains links to TSO and ISPF to perform some of its submission and monitoring functions, but all production control is done within SAS. PCS/AF provides the following features not available in the old PCS:

- A common user interface for all survey production
- Full-screen parameter entry panels with pop-up menu and dialog box support
- Stored parameters for each user/job
- Online session and job submission logs
- Online, context-sensitive help facilities (SAS/AF CBT)
- Interactive testing of both AF and Base SAS applications
- Easy portability to any computing site running MVS/TSO and SAS Version 6.06 or higher

PCS/AF presents the end user with a full-screen, fully-assisted environment for job submission with a combination of features that makes it a dramatic improvement over the old SUPERWYLBUR system. For CPCS data processing managers, PCS/AF frees them from dependence on the outmoded technology represented by SUPERWYLBUR's line mode interface and opens the door to complete hardware independence using SAS as both data processing software and production management tool.

THE DEVELOPMENT PROCESS

When CPCS first expressed interest in developing a new PCS system and asked ARC to undertake the initial investigation, we already had a knowledge of the system's
requirements that was based on years of working with and helping to maintain the old PCS. Therefore, we began by reviewing the software choices. The software alternatives were constrained by what was going to be available at the new computer center (SUPERWYLBUR's availability, for example, was not certain). Of the software packages that could support a flexible job submission system under MVS, only SAS/AF and ISPF were definitely going to be installed. CPCS had substantial existing SAS experience, and the use of SAS software would mean that both the job submission and application code would be SAS-based. Using ISPF panels and TSO CLIST's as the main development tools, on the other hand, would mean developing a whole new base of expertise. Based on these factors, ARC was asked to develop a prototype in SAS/AF. The prototype, which took approximately 3 weeks to develop, was successful, and BLS elected to go ahead with development.

As part of the development process, ARC was asked to work closely with the CPCS production staff to make sure that (1) they had constant input into the design process; (2) they were getting the job submission tools they needed and wanted; and (3) they learned enough SAS/AF and SCL to use and maintain the system. Since ARC's contract was due to expire soon, this last point was especially critical.

Initially ARC believed that because PCS/AF would be written in SAS, a language already well understood and accepted by CPCS management and programmers alike, the job of involving the client in design and implementation would be relatively easy. As it turned out, however, there were several human factors that made the process much more difficult than expected. These factors were: difficulty in communicating the design to the users, CPCS staff unfamiliarity with the role of end user, and lack of user knowledge of the basic TSO/ISPF environment and of interactive SAS.

THE DEVELOPMENT APPROACH

Traditionally, CPCS develops software that is used by other BLS offices. Personnel in these other offices provide requirements and feedback to CPCS during system development. ARC's initial course of action when starting the PCS/AF project was to treat CPCS itself as an end user of this type and to use the same system life cycle approach that was so successful in developing CPI applications. We soon realized, however, that we could not use the standard development approach for this system.

In the first place, the CPCS staff members involved were not used to being on the "client" side of the development dialog. The role of providing requirements, reviewing written specifications and providing criticism and guidance to developers was unnatural to production-oriented people. Because of this, we believed that the feedback ARC would receive when we presented design documents would be limited, and would probably not be sufficient to insure a quality design. In the second place, we knew it was important to generate active participation and enthusiasm for the project, since a tentative or apathetic attitude on the part of the staff would lead to a badly designed system and a wasted software effort. In addition, the impending contract expiration made it critical that CPCS be both willing and able to maintain PCS/AF without ARC assistance.

After some discussion with CPCS management about the problem, we decided to change the development process from a traditional system life cycle approach to an "iterative prototype" strategy in which users would interact with a series of working system models. SAS is an effective tool for rapid prototype development, as the quick turnaround of the original PCS/AF prototype proved. Production staff members were comfortable providing feedback to developers about whether an application worked correctly or was easy to use, and a series of models they could "field test" would allow them to learn and comment at the same time. Since we also needed to include the staff in the actual design and development as much as possible, we decided to involve at least one member of each group in the development of each prototype stage under ARC's guidance. Thus, each survey team would understand the inner workings of PCS/AF and would be able to use and maintain the system without outside assistance.

TRAINING NEEDS AND PROBLEMS

It was very clear that before CPCS personnel could become involved with PCS/AF in a useful way, they would need extensive training. Though they were well versed in the Base SAS product and the macro facility, years of batch processing had presented no need for CPCS programmers and analysts to use any of the interactive SAS Institute products such as SAS/AF and SAS/FSPL. Even when code was developed and unit tested, it was done using SUPERWYLBUR, an editing and file management system that does not permit access to Interactive SAS. TSO, though available, had remained largely unused. It was also clear that some production staff members were unfamiliar with the relationships between SAS, OS JCL, SUPERWYLBUR, and the MVS operating system. The black box approach CPCS had taken to make production smooth, typified by the elaborate SUPERWYLBUR PCS, had the side effect of creating a kind of institutional ignorance of the mechanics behind the production system as a whole. The situation was analogous to the individual who can drive a car, but has no idea how the engine works. If CPCS were to participate in the design and development of PCS/AF, and ultimately to take over the system, they had to understand at least the basics of TSO and SAS Display Manager as well as AF and Screen Control Language. Training in these areas had to be provided.

Unfortunately, all of the resources needed to provide software training -- time, personnel, and money -- were in short supply. CPCS could not afford the expenditure of personnel time or training funds to send the necessary staff members to off-site TSO and SAS training courses. ARC could not provide a generalized training in all these areas because of the limits of its contract and because our personnel were needed for the actual development of the system. CPCS management felt strongly that whatever training was provided had to be directly related to the task at hand. To be effective, the training also had to be reinforced by practice and, if possible, by work on the actual system. In the past, staff members had complained that they forgot the training they had received because they never got a chance to put it to use.

THE "GUIDED DEVELOPMENT" CONCEPT

ARC's solution to the problem was to provide the training as an integral part of the development itself, an approach we have termed "Guided Development". Guided Development is a variation on a more general training concept known as "just-in-time" training. Just-in-time training seeks to teach personnel...
what they need when they need it most, in a manner that integrates the training course with real, existing tasks and job responsibilities. Our Guided Development approach would combine development of PCS!AF and training in TSO, SAS Display Manager, AF, and Screen Control Language. In a sense, this was an extension of the role ARC had played at CPCS for many years, in which we not only developed software, but informally taught many SAS programming concepts and techniques to CPCS personnel.

Using the PCS!AF system design document as a basis, ARC would design a series of short, informal course modules to teach students what they needed to know at least the simpler PCS!AF code modules. For example, after receiving a course on basic AF program entries, the students from a survey staff would begin to lay the groundwork for that survey's main job submission menu. Not only would the training be focused on the exact requirements of the system under development, but the period between course modules would be an extended workshop during which students would become developers, working on portions of the application as far as their knowledge would take them. Throughout the process, ARC would answer questions, critique the work in progress, and make sure the students' programs were conforming to the structure of the program design. In this way, ARC could not only speed the development process, but could take advantage of each team's knowledge of their survey-specific requirements.

COURSE DEVELOPMENT AND IMPLEMENTATION

Our prime concern in developing the course modules was to focus the training on the concepts and skills needed to develop PCS!AF. We accomplished this by using the PCS!AF program design as the basis for both the lecture content and the workshop exercises. First, ARC examined the design and identified the specific SCL elements that were used in PCS!AF. Then, we divided up those elements into basic and advanced topics and sequenced them to produce a course outline. Figure 2 is an abbreviated version of the course outline, showing the objectives for the first two modules.

An important part of each module was its workshop, which was centered around the "workshop version" of the PCS!AF design document. In order to make the lecture material and the workshops follow the development path as closely as possible, we took portions of the actual design and rewrote them as step-by-step instructions for workshop participants. Figure 3 is a section of the workshop versions of the PCS!AF design specification, illustrating how students are "walked through" the process of building an AF/SCL module while actually developing a part of a real application. An important side benefit of this process is that students who will eventually have to maintain a system code the workshop modules by following the instructions. This was done as a quality assurance measure both to test the clarity of the workshop instructions and to make sure that following the instructions with no prior knowledge would lead to a correct result.

In order to keep the number of students small and maximize the amount of attention we could give each student, the survey staffs were asked to select one or two members to send to both the lecture and workshop components of the course series. These selected individuals would become the local PCS!AF experts for their surveys and would be expected to pass on their knowledge to their co-workers. We decided to allow wider attendance at the lecture portion of the modules so that all survey staff members would have at least a basic understanding of the material being presented.

Teaching began in July of 1981. The first module taught basic TSO, ISPF, and SAS Display Manager concepts, and all subsequent modules were on aspects of AF and SCL that were used in PCS!AF. The first SCL module coded by the students was the PCS!AF Session Manager, which had already been coded and tested by ARC. This was done primarily as another quality assurance test of the Workshop Version of the program design. We wanted to be sure that students following our instructions would arrive at a module that was at least similar to the real thing. Each module was taught as a half-day segment, which began with 1-2 hours of lecture and proceeded as quickly as possible to the workshop. Workshops took approximately 2 hours to complete, but were left open-ended so that students with other work commitments could work on them when time permitted.

After the first two modules, subsequent instruction featured less and less lecture and direct guidance by ARC and more self-reliance on the part of the students. Indeed, once students in each survey team began working on their own job submission menus, ARC merely offered technical advice, debugging help, and review to insure that the students' modules would ultimately fit into the new system.

RESULTS

Initially, the course modules were greeted with enthusiasm. The people who attended the first lecture/workshop session were excited about the possibilities that TSO and ISPF offered, and impressed by the convenience of the Display Manager. We had taken care to connect the TSO and Display Manager concepts the students were learning to their SUPERWYLBUR and JCL equivalents, so there was a high degree of comprehension of both how the new tools worked and what they were used for. Enthusiasm was also generated by the sheer novelty of the new software, a factor which we had counted on to make the first module a success. There were some
differences in degree of comprehension of the new concepts, and in the degree of participation (not all teams completed the workshop exercises), but on the whole the first module was very successful.

The second course module was equally well received, but the differences in the teams were becoming apparent. It seemed that a major success factor was the presence on a team of an individual who either had development experience or who had a curious, exploratory attitude toward new computer tools. Those teams with one or more such individuals did substantially better than those without. One sign of overall success was the fact that, after the second module, two of the teams asked if they could move faster: they wanted to begin work on their job submission systems as soon as possible.

The divergence between those teams that were pressing to move ahead and those that were lagging left ARC and CPCS management with a dilemma. Some students had benefited so much from the course modules that they were ready to continue on their own, and some needed much more guidance. Also, given the short time available (the contract under which ARC was working for CPCS as a subcontractor was due to expire in a few weeks), CPCS felt that PCS/AF development had to move ahead, and was not certain that any more time spent on the course modules was cost-effective. Ultimately, CPCS and ARC decided to run the third course module as a workshop only, with no lecture or handout materials other than the specification. Since the third module was to have students begin their own job submission shells, having completed the Session Manager exercise, we were in effect making the students full-fledged participants in development.

At this point, ARC's prime contractor lost its contract, and ARC was unable to make arrangements to continue as a subcontractor under the new prime — a possibility that had been foreseen but not expected. Time became very critical, as ARC had to begin moving people off the contract immediately, and only six weeks remained for any of our personnel. PCS/AF development became a crash program, as we struggled to turn the system and its documentation over in a state that would allow CPCS to complete it. Efforts to "guide" the students in their efforts, though not abandoned, were reduced to weekly meetings and rather hurried one-on-one conferences.

Given these unfortunate circumstances, ARC and CPCS management were both surprised and pleased when we found that, by the end of the contract, two of the four teams (including one we had labeled as "lagging") were well on their way to taking over their portions of the PCS/AF system. Some of these team members, in fact, were helping ARC work on some of the more complicated utility programs that supported JCL file tailoring. A third team was beginning work on its job submission shell, which was more complicated than the others. The fourth team had been diverted away from PCS/AF by other pressing system problems. The concepts and skills taught by the workshops, augmented by practice, were taking hold. The just-in-time training had indeed occurred just barely in time to save the software development effort.

CONCLUSIONS

ARC was tasked with the completion and turnover of a major SAS/AF system, PCS/AF, in a very short time. Since we would be unavailable to assist our clients in running and maintaining that system once it was turned over, we also had to provide them with the knowledge they needed to be the system's true owners. To speed development and bring the client's programmers "online" quickly, ARC employed a concept that is relatively new to SAS application development and software training. Known generically as "just-in-time" training, the Guided Development method ARC used at CPCS combined software instruction with direct, hands-on experience in actually building a system. We believe that the focused training, and the sense of involvement with the new product it fostered, gave our client the extra help they needed not only to take over a complex software system, but also to learn some powerful new tools for system development.
FIGURE 1: A FUNCTIONAL OVERVIEW OF THE PCS/AF SYSTEM

[Diagram showing the functional overview of the PCS/AF system with various nodes and flows described in the text.]
FIGURE 2: GUIDED DEVELOPMENT COURSE OBJECTIVES

Workshop #1: The PCS/AF Environment

Goal: To familiarize students with TSO, ISPF, and SAS Display Manager

Objective 1: Logon to TSO, and use TSO commands to examine a file and check on the status of a batch job.

Objective 2: Invoke ISPF, learn to browse and edit files from ISPF; and use ISPF utilities to create (allocate) a partitioned data set (PDS) and a SAS data library.

Objective 3: Invoke SAS from ISPF and from native TSO; use Display Manager to create, run, review, and save a program; use SAS commands and windows to identify, use, and examine the contents of a SAS data library.

Objective 4: Invoke ISPF from inside the SAS Display Manager and use it to identify the name of a SAS data library.

Workshop #2: SCL Basics

Goal: To teach students the basic components of SAS/AF and Screen Control Language

Objective 1: Execute PROC BUILD to create the AF PROGRAM entry that will become the Session Manager.

Objective 2: Lay out the screen for the Session Manager as directed in the PCS/AF Program Design (Workshop Version).

Objective 3: Define SCL variable names and attributes for the PROGRAM entry.

Objective 4: Define the three basic sections of the SOURCE portion of a PROGRAM entry.

Objective 5: Use SCL programming statements and functions to code a preliminary version of the Session Manager, as directed in the PCS/AF Program Design (Workshop Version).

Objective 6: Compile, review, and test the Session Manager PROGRAM entry.
FIELD ATTRIBUTES (ATTR)

Each screen variable (defined with an & in front of it) on the Display screen must be given the proper attributes.

1. Define the attributes for each field. Each field has a separate "page" for its attributes. When finished with the attributes for one field, enter DOWN at the command line to go onto the next.

For each field, define field attributes as follows:

&DATE 12 bytes, numeric, formatted (WORDDATE12.), protected, no alias.

&TIME 8 bytes, numeric, formatted (TIME8.), protected, no alias.

&USER 6 bytes, character, no format, alias USERID.

OPTION variables - all of the following variables will have the same attributes except for their initial values and list of valid values.

Basic attributes: 3 bytes, character, protect set to initial, caps on, and autoskip on.

Field aliases and value lists:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Alias</th>
<th>Initial Value</th>
<th>List</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;CA</td>
<td>CAS</td>
<td>CAS</td>
<td></td>
</tr>
<tr>
<td>&amp;HO</td>
<td>HOU</td>
<td>HOU</td>
<td></td>
</tr>
<tr>
<td>&amp;CE</td>
<td>CES</td>
<td>CES</td>
<td></td>
</tr>
<tr>
<td>&amp;IN</td>
<td>IND</td>
<td>IND</td>
<td></td>
</tr>
<tr>
<td>&amp;AP</td>
<td>APP</td>
<td>APP</td>
<td></td>
</tr>
<tr>
<td>&amp;TS</td>
<td>TST</td>
<td>TST</td>
<td></td>
</tr>
<tr>
<td>&amp;RE</td>
<td>REV</td>
<td>REV</td>
<td></td>
</tr>
</tbody>
</table>
FIGURE 3, CONT.

General Attributes (GATTR)

Certain general attributes can be defined for the SCL program, including the location of a customized PF key set and the location of Help files (if any). In this section, you will tell the application where to find the PF keys set in the previous section:

1. Open the GATTR window by entering:
   
   GATTR
   
   on the command line of the ATTR or DISPLAY window.

2. Enter the PF keys catalog entry name:
   
   AFLIB.PCSAF.SESSION.KEYS
   
   on the appropriate line of the GATTR window.

3. Save the work done thus far using the SAVE command.

4. Close the GATTR window using the END command.

PF Keys:

In addition to the choice options on the screen, the user may press one of three PF Keys. PFK1 will be defined as general 'online' HELP, PFK3 will be defined as QUIT (END), and PFK12 will be defined as TSO IOF.

1. Open the KEYS window by entering

   KEYS
   
   on the command line of either the DISPLAY or ATTR windows.

2. Replace the text for PFK1 with HELP

3. Replace the text for PFK3 with: END

4. Replace the text for PFK12 with: X'IOF'

5. Replace the text for all other keys with blanks

6. Save the KEYS window using the SAVE command as:

   AFLIB.PCSAF.SESSION.KEYS

7. Close the KEYS window by entering

   END
   
   on the command line.

SCL Program Code (SOURCE)

INIT: Initialize the session for the user

1. Create the SCL variable USERID in which the user will enter his userid.

2. Create the SCL variable P_ACTION and assign it an initial value of '6' ("Logon").

3. Call the Session Manager System log display (PCSAF.UTILITY.SESLOG.PROGRAM) with the parameter P_ACTION
FIGURE 3, CONT.

4. Assign the SCL field variable DATE today's date.

5. Assign the SCL field variable TIME the current time.

6. The MAIN: block should execute whenever the user presses the ENTER key. Use the SCL CONTROL statement to indicate this.

MAIN:

Choose System Option(s)

Options available to the user will be selected through a 'point and shoot' type of menu. Each available choice will contain the first three characters of the option.

1. Set the SCL variable OPTION to the value of the CURWORD() SCL function.

2. Process the user's choice:

   2.1 If OPTION = 'APP' then do the following to call Application Builder:

      2.1.1 assign the SCL variable P_ACTION the code '2', ("Building Application").

      2.1.2 Call the "dummy" application display (PCSAF.UTILITY.DUMMY.PROGRAM) with the parameters USERID and OPTION.

NOTES

1 Trimble, J. Harvey: Can SAS Replace Traditional Programming Languages? SUGI Proceedings, 1982

Mopsik, Judith H.: SAS Can Replace Traditional Programming Languages, SUGI Proceedings, 1983


ACKNOWLEDGEMENTS

SAS, SAS/AF, and SAS/FSP are registered trademarks of SAS Institute, Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

The authors would like to thank Steve Wright, Division Chief of the Division of Consumer Prices and Consumption Studies of the U.S. Department of Labor, Bureau of Labor Statistics; and Eli Honrickson, Veronica Fasullo, and Doni Lazenby of Atlantic Research Corporation, Information Systems Division for their contributions to both the PCS/AF system and to this paper.

The authors can be contacted at:

Atlantic Research Corporation
Information Systems Division
1301 Piccard drive, 2nd Floor
Rockville, MD 20850
(301) 258-5300