USING SAS SOFTWARE WITH TSO/SPF TO MONITOR NETWORK AVAILABILITY

ANITA BLITZ, EQUITABLE LIFE ASSURANCE SOCIETY
GAIL LEVINE, EQUITABLE LIFE ASSURANCE SOCIETY

ABSTRACT

This paper will describe the development of a system to track problems related to various hardware components in a nationwide on-line health claims payment system.

The Network Availability Logging and Reporting System (NALARS) was developed to fulfill a need for accurate, complete information on vendor hardware problems. It was done in two stages. We will discuss the reasons for the two-part development method, the extensive error-checking, a high degree of user-friendliness and the use of SAS(R)[1] in accomplishing these goals.

NALARS is used by two distinct groups of people--network control operations staff and network management personnel. The needs of both groups and the solutions to their respective problems will be explained.

To accomplish our goals of ease of use and assurance of accuracy, we employed many special features of SAS(R). The use of these features will be described as will the interface between SAS(R) and TSO/SPF.

Finally, we will discuss the reactions of our users to the System as well as future plans for enhancement.

BACKGROUND

The Performance Information Management System (PIMS) group is responsible for collecting, analyzing, databasing and reporting on performance statistics related to the EQUI-CLAIMS system. This is a nationwide health claims payment system administered by the Equitable Life Assurance Society.

EQUI-CLAIMS is actually 4 CICS systems running on two 3081K IBM processors. Currently, both processors are located in one data center and the telecommunications network is controlled from that site. There are approximately 3500 terminals & printers, with their associated controllers and circuits spread among 40-50 field offices. The present network configuration allows up to 23 devices on one controller. Current statistics show a workload level of about 300,000 CICS transactions per day per system with an average arrival rate during peak periods in the neighborhood of 10 to 11 transactions per second. At this level of activity, lost work hours due to an out-of-service component can quickly add up--especially if the component is a controller or a circuit, as opposed to an individual CRT.

PIMS was originally formed to aid the performance analysts in digesting and summarizing the voluminous data coming out of various system monitors, e.g. Performance Analyzer II, RMF, SMF. We needed a means of culling only the data of interest to us, summarizing it and saving it for later analysis. This analysis would be used to spot trends, predict future hardware needs and guide us in fine tuning the system for better response time. To this end, we designed a system using SAS(R).

SAS(R) was chosen because it was easy to expand as new needs became evident and it would be easy to develop ad hoc reports without a long lead time. We established separate SAS(R) databases for each CICS system. In addition, we created program, macro and format libraries to support the system.

DESCRIPTION OF SYSTEM

Problem Definition

Network management was having a difficult time monitoring the various hardware components of the network. Operations personnel were supposed to write a 'trouble ticket' whenever a device was reported to be working improperly or out of service. They were then supposed to contact the vendor responsible for servicing the component and track the problem until it was resolved.

They were also responsible for writing daily logs from these trouble tickets, summarizing the status of the problem. These reports were for their own use but were also vital to management in their efforts to spot problems with an individual component, particular locations and/or specific vendors. Lastly, management used them to compile a monthly report of hours lost in each field office due to network component failure.

This manual system was, for many reasons, not working adequately. Operations personnel were often too busy to stop what they were doing to write a trouble ticket. The tickets had a way of getting lost or the person handling the problem would forget to note when a problem was resolved. In addition, writing the reports was time-consuming both for operators and management. Information on the status of a problem was not readily available without going through a pile of hand-written logs. Spotting trends and specific problem areas was difficult if not impossible.

In order to alleviate these problems, network management asked us to investigate a system whereby the information could be easily recorded directly into computer storage. The specific goals were:

1. An easy-to-use accurate method of recording data in an accessible computer-based history file.
2. Daily reports to be used in tracking outstanding problems.
3. The ability to get reports from the history file to substantiate poor vendor service, pinpoint faulty components and analyze performance by field office.

[1] SAS, SAS/FSP, and SAS/IMS-DL/1 are registered trademarks of SAS Institute Inc., Cary, NC, USA. SAS/GRAPH, SAS/ETS, SAS/OR, and SAS/REPLAY-CICS are trademarks of SAS Institute Inc.

4. Readily available answers when a field office calls to find out the status of a component in need of repair.

Requirements & Limitations

In order to be successful, the Network Availability Logging and Reporting System (NALARS) had to meet several criteria:

1. Expediency - something had to be done immediately to begin logging data and building a history file.
2. Ease-of-use - the data entry system had to take a minimum of the operators' time and be easy for non-technical people to learn.
3. Accuracy - unless information was recorded correctly, it would be useless to management. In addition, we had to ensure that data would not be accidently lost.
4. Expandability & Flexibility - to satisfy the expediency requirement, the system would be developed in 2 main phases: Logging and Summarizing. We needed to design it in such a manner that we could postpone the design of Phase II until after Phase I was successfully in place.

Phase I - Logging

Logging is in the province of the operations area and they have special needs which differ from management's needs because of the nature of their jobs.Network Control operations personnel are responsible for keeping the telecommunications system running smoothly. It is their job to take calls from field offices when they have hardware proolems, call the appropriate vendor for service, and keep track of the problem until it is resolved. They are also constantly switching lines to provide coverage for an office if one of its controllers is cown or it simply needs help from another office. They are constantly busy and time is at a premium. In addition, they are not highly trained technically and do not have programming skills. Any system designed for their use had to be fast, easy-to-use and have as many validation procedures as possible.

One solution would have been a full-screen data entry system with prompts and tightly-defined fields. Each screen could have held the information for one trouble ticket. We could have accomplished this through a custom-designed SPF screen. However, it was also necessary to be able to retrieve a particular 'ticket' in order to update it, add comments to it, answer questions about it or just review it. Custom screens could not be updated under the version of SPF (Release 1) we were then using.

We probably could have accomplished this goal with SAS/FSP(R) but we did not have this product in house. Not wanting to delay the project (expediency requirement) while we investigated this product and then waited for it to be installed, we decided to compromise. We provided a TSO data set with a pre-written screen calling for specific information to be entered by the user (Network Control Operations) during the online day.

The following data is included on the trouble ticket form to be filled in by the operator:

- a ticket number consisting of the block number (MMDD) followed by a unique number for each ticket created that day.
- the field office, including site contact and phone number, where the failing component is installed.
- the identification number of the component, inclucing a code for type of device.
- the name and phone number of the vendor responsible for service.
- the date and time the component was reported to be in need of repair and the date and time it was restored to service.
- an 8 line free-form area for comments.

Each night, in a batch job, the information entered during the day is analyzed and either written back out to the TSO data set or recorded in a permanent history file. At this time, the cumulative number of hours this device has been out-of-service is entered on the ticket and appears with the data entered by the operator. Blank forms are provided in a separate data set which the operator can append to the end of the active one when needed.

Since this is not an interactive on-line system, it does not provide immediate feedback to the operator if he makes an error in recording data. This is the prime limitation in using SPF/EDIT instead of a full-screen data entry package. The other major problem is the lack of protected fields.

This could possibly result in erasing field labels and/or information (such as lost hours) which is supplied by the system or another operator. We will discuss our use of special SAS(R) features to solve this problem further on in the paper. Although this compromise does present problems, it has the benefit of faster processing for the operator and better use of his or her time. And, although it could compromise the requirement of accuracy, most operator errors are flagged during the batch run and the staff is informed of them the following day. No 'ticket' with a flagged error is ever written to the permanent history file until that error has been corrected.

The system provides three daily reports for the use of Network Control. These are "Open and/or Invalid Tickets", "Components Out of Service Longer Than Normal" and "Log Tickets Closed During the Day". The first report points out items that are still out-of-service as well as those which contain a data entry error. The second report flags problems that have been unresolved longer than a predetermined "normal" threshold thereby demanding immediate attention. This threshold can be different for each device type. The last primarily serves as a check against losing an entire ticket. Since each ticket is

numbered, a comparison of today's reports with yesterday's should account for every ticket either on the "Open..." report or the "Closed..." report.

A history data set is provided to keep a record of all resolved problems. Until Phase II, this information was only available as a non-selective print-out of all the records in the file. In Phase II, programs would be developed to subset the file in various ways.

We decided to keep this as an OS file instead of adding it to our SAS(R) databases. The primary reason for this decision was space. We felt it was important to allow the operators a kind of scratch pad on which to record information about the problem as it was being resolved. If a problem was particularly difficult to resolve, there might be comments added from time to time. In other cases, the comments field might be only a few words. We have allowed up to 626 characters for comments.

Therefore, a variable length record seemed to be called for. SAS(R) does not allow for this so we decided to store the data in an OS file and convert it to temporary SAS(R) data sets for reporting purposes only. Since, in only a year and a half, over 7000 records have been written to this file, and most of the comments are less than 50 characters, we have saved a considerable amount of disk space by storing the data in this way.

Phase II-Summarizing

This part of the project is primarily for the benefit of network management. Management in this case is spread through more than one department and location. Again, the users are non-programmers. The need here is for information retrieval--not data entry.

There were two major information-gathering functions being performed by network management. First, they were manually producing a monthly report of line outages summarized by field office. This was extremely time-consuming and probably inaccurate. It depended for its information on the daily hand-written logs kept by network control operators. As part of NALARS Phase I, we had provided a TSO CLIST to generate a listing of the history file starting with a user-specified date. Therefore, accuracy improved but preparing the report was still an effort requiring a good deal of time. The information in the print-out still had to be summarized by field office, organized and either hand-written or typed.

The second information need was ad hoc. A field office might be complaining about poor service, necessitating a detailed report on outages in that office, or management might want to study the performance of a given vendor in terms of the length of time between reporting a failure and resolving the problem.

The first need (the monthly report) could be satisfied by incorporating monthly processing in the batch job which updated the history file each night. The report, "Monthly Line Outage Hours By DBO", provides a listing for each field office for the preceding twelve months. The number of work hours lost due to equipment failure is totalled by month along with the number of separate occurrences. An entry is made for each office, for each month, regardless of whether any outages occurred.

The second need required a user friendly interface between SAS(R) and TSO/SPF. It had to allow the user to specify conditional parameters for a given run of the job and it had to be accessible to several different users in different locations. These requirements were easily satisfied through a custom-designed SPF background job submission screen. Background was chosen as opposed to foreground, so that the user could submit the job and leave the terminal to return to his or her other duties. Since the desire was for a written report instead of a screen display, there was no reason to make the user wait while SAS(R) processed the request.

To date there are 3 different reports available to the user through TSO: "Abnormal Outages," "Outages by DBO", and "Vendor Component Outages." We provided an additional option on the primary menu of each TSO user who would need to access the system. This takes them to a secondary menu from which they choose the report they want. This screen also allows for each user to have a different job card and print destination. Several reports can be submitted in one job. Each report has its own screen in which the parameters for that report can be entered. The SPF program incorporates the values into symbolic variables on the JCL control cards. These are read by the SAS(R) program which tests them against data values in the history file.

The "Abnormal Outages" report allows the user to specify the starting and ending dates of records to be pulled from the history file. It also allows them to choose one particular field office or all field offices. In addition, it permits setting of thresholds for each device type. In other words, it will only select a record from the history file if it falls within the specified time interval, refers to a component in the stated office and indicates lost hours greater than the threshold set for that type of device. The report then prints the information in each trouble ticket record grouped by vendor and within vendor by DBO (field office).

The "Outages by DBO" is a simpler report. Here the user can specify a time interval and either one particular office or all of them. This report contains no vendor information."Vendor Component Outages" is similar to the above report except that it contains vendor information.

All of the reports contain the trouble ticket number. If more details are needed, the history file can be browsed on TSO/SPF and a 'find' done for that ticket number. This will bring up the required record for quick review.

SAS TECHNIQUES

In this section of the paper, the SAS(R) techniques used to implement these tasks are described.

The SAS(R) program which reads the daily log each night is responsible for performing many of the functions described above. It checks each log entry for errors, calculates the number of lost hours, writes records for components which have been repaired to the permanent OS history file, tests for inclusion in any or all of the three daily reports, and, once a month, triggers the monthly 'Line Outages' report. In addition, provisions had to be made so that as field offices opened or closed, information about them could be added without having to alter the program.

Inputting the data

As mentioned earlier, the log entry is on a pre-written screen consisting of 23 lines. The information is input as a multi-line record with 19 variables (including the variable length comments field). This method of inputting the data was chosen in order to have, in one observation, all information pertaining to a record, without either having to retain variables or having to verify record keys for each variable.

Calculation of lost hours

The number of lost hours for a given component is considered to include only those hours during which the office in which it is installed is open for business. Therefore, the calculation of this statistic includes the time between malfunction and office closing on the first day, the time between office opening and repair on the last day and all days in between. Saturdays and Sundays are not included unless the office was open on that day.

An office's working hours are determined by the CICS system in which it resides. At present, all systems come up at the same time but are brought down at different times. These hours can change dynamically. Therefore, in order to avoid changing the program every time on-line hours are adjusted, a format was created to associate a given system with its closing time. We also created a format to associate a given office with its CICS system. The program uses this information by testing the system-formatted value of the office and then using the end-of-day-formatted value of the system. The PUT function is used for this. The MDY function is then used to convert to a SAS(R) time value and any necessary subtractions are performed.

Because problems are not always entered immediately (down times or restored times are sometimes entered days after the incident), the lost hours are recalculated each night. This also eliminates the concern that the previously calculated amount could be accidentally erased or changed during the day. A DO WHILE loop is used. It is executed once for each day between the date the component went down and either the current date or the date the component was restored, accumulating the total number of lost hours.

These figures are compared, by device type, to predetermined thresholds. If the component is still not working and its lost hours are greater than the applicable threshold, it is listed in the status report, "Components Out-of-Service Longer Than Normal."

If the component is back in service, the record is written to a permanent history file along with the total lost hours. If the component is still out of service the entry is written back to the daily log in its original 23 line form but with the number of lost hours entered. Until it's repaired the new lost hours will be calculated each night and rewritten to the log.

Comments

An important element in the daily log is the detailed description of the problem. While the component is still out of service this information is used when speaking to the vendor to explain the malfunction.

After the component is restored it provides a history of the types of problems which have occurred. To allow a maximum of flexibility, eight lines are provided in the log for these comments. It was decided that on the printed status reports, each problem would be restricted to one line (133 columns). This meant that only a portion of the comments could be included. The permanent history file, however, would need to contain the complete comment field, regardless of the number of lines necessary.

Information can be entered anywhere on the eight lines, using as many lines as necessary and often leaving many blanks between sections of comments. In order to include as much information as possible on the status report, and save space in the history file, each line of the comments is treated as a separate variable and read with the VARYING format. The actual length of each comment is determined using the LENGTH function. It is then written to the file, the column pointer is incremented and the next comment line is concatenated to it. For the reports this is done until the number of columns used is greater than 133 or there are no additional comments. For the history file, the 133 column limit is not invoked.

Error checking

Each entry in the log is checked for as many possible typographical errors as possible. Every field is examined for a wide variety of potential problems. While it is not possible to catch all errors the majority are caught using this method.

The VERIFY function is used to check the alphabetic and numeric fields. The SUBSTR function allows us to check part of each field. For example, the ticket number should be a 6 character numeric field, where the first two numbers represent the month, the

second two the day. We first verify that it is numeric, then use the SUBSTR function to make sure that the first two numbers are not greater than 12, and the second two numbers are not greater than 31. We also use the LENGTH function to make sure that the length is not greater than 6.

Each field is checked to verify that only valid component and office codes are entered, that the date the component went down is not greater than when it came up, and that the time the component went down or came up is not later or earlier than the closing and opening times, respectively, of the office. If a field is found to be invalid, it is overwritten by asterisks, an invalid flag is set and the entry is re-written to the log file. This is achieved by reassigning the value of the invalid variable to asterisks, using the REPEAT function (VARIABLE=REPEAT('*',x) where x is the length of the variable). An entry is also written to an error report.

The next day the report is examined and the errors corrected by network control operators. If the component was repaired, the next night's run will enter it in the history file.

Daily reports

As described above, there are three daily and one monthly report produced by the batch program.

A number of different people receive the reports, in slightly varying formats, to serve their needs. Therefore, a record can appear in several reports. As each entry is processed, it is written to the appropriate SYSOUT file(s) for the report(s) in which it should be included. The JCL for the job has a DD card for each SYSOUT file with a JES2 statement directing the report to the correct print destination. "IF" statements are used to determine whether a record should be included in a report and/or written to the open or closed file. It is written using FILE and PUT statements. In addition to the DD name the FILE statement includes a HEADER= option. All of the headers are defined within the program.

If an entry is still open, it needs to be written back to the original OS file. Since SAS(R) does not allow in-place updating of an OS file, the OS file is written to a temporary data set. After the data step is complete, and all open or invalid tickets are in the temporary data set, PUT _INFILE_ is used to write all the records from the temporary data set back to the original file.

Monthly report

PROC TABULATE was chosen to produce the monthly report because it automatically calculates the number of observations and totals the lost hours for each month within each office. In addition, its output resembled the format in which management had been producing the report manually.

The conditional execution of the monthly report is triggered by comparing the current rundate to the last date the job was run using the MONTH function. If the last date was a different month than the current rundate, a temporary SAS(R) data set is created as a control card, with the last day of the previous month as the end date. The INTNX function is used to establish the startdate of the year to be reported. In this way, the report will always contain the most recent 12 months even if it straddles 2 years.

A temporary OS file, with the DD name TABULATE, is then opened and the statement "%INCLUDE PROCTABS" is written to it. This is not done unless the afore-mentioned SAS(R) control card had been created. PROCTABS is the DD name for a partitioned data set which includes the code for the PROC TABULATES. If it is not a new month and there is no SAS(R) control card, the program will end. However, if it is a new month, at the command "%INCLUDE TABULATE", " %INCLUDE PROCTABS" is substituted, which becomes an executable line of code. (This code was written prior to the availability of SAS82(R) and we are now investigating if the new macro language will be a more effective method of handling this situation. For a complete description of the conditional execution of code see Don Henderson's, SAS TUTORIAL: CONDITIONAL EXECUTION OF DATA AND PROC STEPS, SUGI '82, pp. 787-791.)

PROC TABULATE provides many features which had to be carefully manipulated when inputting the data. Because the history file is not in any meaningful order, and there is not necessarily an entry for each month for every office, a paramount concern was insuring that the table have an entry for each office, for every month, regardless of the number of lost hours. The MISSING option would still not include offices which had no lost hours during the preceding year.

To maintain the correct order, an entry is forced for each month for each office. This is accomplished by reading a list of all offices in a DO UNTIL loop and outputting one observation for each month from the start date to the end date with all other variables having missing values. The ORDER=DATA and PRINTMISS options are used.

An interesting note is that the clarity of the table was directly affected by the number of lines used to form the squares. If the default was used there were so many lines that it was difficult to read the table. After experimenting, we used the FORMCHAR option to get solid vertical lines and dots for the horizontal axis.

Format library

A special feature utilized extensively within the program is the ability to write your own SAS(R) formats. These formats are stored in a partitioned data set and called into the program using SASLIB.

Formats are used in two main ways (in addition to defining formats for value labeling). The first use is to allow the

addition or deletion of offices without having to change the coding of the program. Because these lists are in a format library we can change them the same day the office opens without rewriting the program.

The second, and most important use, is to assign one variable based on another variable. Each field office is entered in the log as a three letter code. From this code, the format tables allow us to identify it with one of four CICS systems, locate the geographic region it is in, determine the date it was first opened or closed, and retrieve the hours of the day that the office is on-line. The PUT function can then be used to create a variable by changing the DBO code to the required format.

EVALUATION AND PROBLEM RESOLUTION

Phase I of NALARS has been in operation since April 1, 1982. There were, as expected, problems at the beginning but, with minor changes, the system has proved almost mistake-proof and of great benefit to all who use it.

One early problem, which we anticipated, was that the relatively inexperienced network operations personnel would erase random lines in the TSO data set. This would cause the wrong data to appear on a line. Our error-checking procedures were entirely effective in catching this. Of course, we ended up with an asterisk-filled ticket. This was very difficult to reconstruct and we had to help at first. However, the operators had been instructed to make frequent hard copies of the data set and this proved to be very important. The difficulty in reconstruction led the operators to be more careful in their use of the system.

Another problem was an inconsistency, from operator to operator, in the 3-letter codes used for each field office. The original version of the program did not use the 3-letter codes for calculation of on-line hours so there was no problem during job execution. However, the history file could not be trusted to yield all records for a particular office. To correct this problem, a format was created listing each valid code and a validation procedure was added to check this field on the log ticket. Now the DBO (field office) had to be not only all alphabetic but a recognizable combination of letters.

By the end of the first month or two, calls for help were very infrequent and usually the result of an office or new device type having been added without informing us. Now, the operators are learning to ask us to add these items to our tables before they cause errors. Feedback has informed us that the system has saved them time and that they make extensive use of the daily reports in their efforts to resolve problems. As new operators have been hired, the present staff has been able to train them to use the system without our help or even our awareness. This testifies to the ease-of-use of the TSO log.

Phase II has been in effect for only about 6 months. As stated earlier, the primary beneficiaries of this portion have been network management. It has fulfilled their expectations and saved them a great deal of time. They no longer prepare any reports by hand and they have been delighted with the ease and speed with which we can make changes to the formats of the on-request reports.

Although they have not asked for additional reports, the structure we have set up will allow us to add them in a timeframe of days or weeks instead of months.

FUTURE PLANS

Since NALARS was developed before SAS82(R) was available to us, it does not make use of the new MACRO language. We plan to investigate the possibility of enhancing or streamlining the programs using this capability.

Also, we are in the process of installing a network hardware monitoring system called NET/ALERT at our data center. We intend to explore the use of data from that system as automatic trouble ticket input. This may necessitate a major change to the logging function.

Recently, ISPF was installed on our mainframes. We have converted the on-request report screens to the new format but they do not take advantage of any new capabilities. We intend to explore the possible benefits to be gained in a more thorough conversion.

Lastly, we are going to be installing another data center. We will have to consider the adjustments necessary to monitor the network from two sites.

We are confident from our past experience with SAS(R) that the structure we have established will smooth the necessary transitions.