ON MIGRATING A SAS-BASED DATA BASE TO A PRODUCTION SYSTEM

Thomas G. Kryzak

New York City Health & Hospitals Corporation

ABSTRACT

A non - EDP unit of the New York City Health and Hospital Corporation developed prototype SAS Programs to create a consolidated Clinical/Financial data base for 11 acute care hospitals. A review of external vendors' data base systems indicated that in-house development would produce a system with greater functionality at less cost.

It then was necessary to convert the prototype programs into a production system to be run on a regular basis by the EDP Department. This paper discusses the characteristics of the data, the reasons for creating a production system and the considerations involved in migrating to a full production environment:

The reasons for creating a production system involved:

1) size of the data base
2) Computer resource requirements
3) The local processing environment
4) System continuity
5) System integrity

The implications of so doing were:

1) Documentation
2) Control Reports and Data Edits
3) Program Staging
4) Operator Instructions
5) Testing

INTRODUCTION

The New York City Health and Hospital Corporation (HHC) operates eleven acute care inpatient facilities. Clinical and financial data are maintained through a shared services vendor. These data were accessible only on a hospital specific basis and only with difficulty and much expense. A changing reimbursement environment required enhancements to each hospital's database. And, at the same time, HHC decided that it would be desirable to consolidate the individual databases into one large corporate data base.

The size of this database can be appreciated because it would have to include at a minimum 1 demographic record, 20 service records and 10 diagnosis and medical procedure records for each of the approximately 250,000 patients we serve in a year. Moreover, these records have to be categorized to indicate into which of some 500 types of disease or condition the patient has been placed (this categorization is into Diagnosis Related Groups, or DRGs).

Several private vendors offer systems that accomplish the categorizing and also create data bases. These systems were reviewed and evaluated for use at HHC. While the review was proceeding, a small unit, outside of the EDP department, but possessing SAS programming skills, began to develop a prototype data base system. The original intent was to establish benchmarks of what was feasible to expect. However, the systems investigated proved to lack desired features and were costly.

Based on the development work already accomplished, it was decided to create the corporate data base using SAS as the programming language.

SAS had been introduced to HHC about one year prior. We, therefore, decided that the database would be stored as a collection of SAS data sets which were initially generated by the prototype programs. However, it was further determined that the data base would become a regular production system rather then continuing to be produced on an ad hoc basis.

There were several reasons why it was desired to migrate the data base to a production system. These include:

1. Size
2. Computer resource requirements
3. Processing environment
4. System continuity
5. System integrity

SIZE

The data base is of considerable size. It includes data on 382,000

patients, has 8.5 million records and occupies 2} 3380 disk packs.

COMPUTER RESOURCE REQUIREMENTS

The first time conversion of a system makes unusual demands on computer resources, primarily due to the need to backload records. More important are the ongoing, cyclical demands placed on the system. Our data base is updated quarterly. i.e. every 3 months. Each update cycle involves 15 batch jobs, 2} hours (150 minutes) of CPU time. THe batch jobs take 13 throughput hours.

PROCESSING ENVIRONMENT

SAS is available to non-EDP department users through an Information Center. THis means that this resource is oriented towards:

1) limited data input sources,
2) Data analysis rather then data management.
3) Hard copy report output

This orientation in turn, entails the limited availability of hardware resources and JCL transparent to the user, hence highly inflexible for the user whose purposes are beyond the intended scope of the Information Center.

SYSTEM CONTINUITY

The data base has quickly demonstrated that it is a major corporate asset with useful potential that cuts across virtually every operating unit. Such a resource should not be the casual property of one or a few individuals, but has to be set up to have an existence that is not closely linked to and dependent upon those individuals. By becoming a production system, the data base becomes the property of the organization with a life span that extends beyond the tenure of any individual in th same manner of the Corporation payroll or accounting systems.

SYSTEM INTEGRITY

As valuable assets the data base itself and its associated programs have to be protected from accidental or purposeful misadventure. Moreover possible access to confidential and/or proprietary information requires careful screening procedures and security systems that permit only authorized users to view the data. There were several considerations in transforming our ad hoc programs into a production systems. There areas are:

1) Documentation
2) Control reports & Data Edits
3) Program Staging
4) Operator instructions
5) Testing

How these areas have to be handled differently in a production environment than an ad hoc one will be discussed. One overall difference is that the running of the system will no longer be the direct responsibility of the owners of the system,i.e. the immediate feedback that comes from great familiarity with the input data and the expected outputs, will no longer be immediately at hand.

DOCUMENTATION

Great care has to be taken with a system intended for the production process with regard to the way the programs are documented. It has to be assumed that the people who will have responsibility for the data base production will not know the data nor the programs. Therefore, the documentation has to be explicit in explaining what the purpose of the program is, the flow and relationship of intermediate steps, and the structure of all input, intermediate and output data files. We documented our system by providing for each major segment:

1) an overview flow chart
2) a statement of purpose
3) a structural decomposition diagram
4) data set layouts for each input, output and intermediate data set

This effort was greatly facilitated by the self-documenting features of SAS such as PROC CONTENTS for identifying the variables in data sets, the source statements used to create those data sets, and the history of prior data sets that contributed to the one in question.

CONTROL REPORT & DATA EDITS

For the experienced SAS user, the log presents enough information to indicate the outcome of the job even in the absence of PROC outputs. When the job, however, is rum by others, more explicit data is necessary. In making the

transition to a production system, it was necessary to program control reports, which indicates the number of input records and the disposition of those records so that there will be a "audit trail" to account for all input. Moreover, the control reports show how many records fail the programmed edits and provide a check of the quality of the incoming data. Too high a failure rate suggests that the data base that would result would be of questionable integrity and should be further investigated by those who 'own' the data base.

PROGRAM STAGING

A large scale data base like ours entails significant commitment of computer resources. In a production environment, there are many competing jobs. The question of how to add a new system of jobs to this environment needs careful planning. SAS facilities this process because it is possible to make small scale trial runs for the system, then use the information that SAS returns in the log. For each DATA & PROC step, the log indicates how much core and CPU time was used. THese values can then be extrapolated to correspond to the full size system.

OPERATOR INSTRUCTIONS

Similarly, explicit instructions to the computer operators have to be provided since they will be running the system with little or no intervention on the part of the system's developers. Each installation has its own standards for its 'run book' which have to adhered to. In our environment, we had to provide complete directions on which programs to run, which input and output tapes to mount and what to do if the job failed to complete successfully. In addition, we separated the programs the extracted data from external sources from the rest of the programs that processed that data and maintained the database. We were then able, because of the control reports produced in the data extraction steps, to have the operators evaluate the quality of the in-coming data and based on the standards in the run book, decide whether to proceed with the rest of the program.

TESTING

Again, because of the reliance that many corporate and hospital users will be placing on the information in the data base, it is imperative that the programming for the data base be tested to insure that the data are accurate and complete. Moreover, because of the security procedures the programs had to be checked and verified before being transferred to the production environment.

This testing involved several phases. The first was unit testing which involved testing each program individually. Then, since this is a system involving sequentially running several different programs, we had to do string testing which merely means running the several programs together to insure that there were no cross-program inconsistencies. These two phases of testing were done using small test data sets for reasons of speed and efficiency. Thirdly, there was volume testing with the full set of source data to ascertain job requirements and run scheduling. Lastly, there was extensive testing of the data output to assure ourselves that what we were capturing into the data base was accurate and complete. This phase was accomplished by running the system on pre-defined input data where the expected output was already known. We were also able to create the various error conditions that might be encountered, even if only extremely rarely, to check the edit routines that were built into the programs. We also compared the output using 'live' source data to make certain that we were accurate and complete. This process was facilitated by using the LIST facility of SAS to 'dump' the source data.

For more information, contact the author at:

New York City Health & Hospitals Corp.
230 W. 41 Street
New York, New York 10036