AUTOMATIC TRANSMISSION AND CAPTURE OF MEDICAL DATA
FROM A MUMPS SYSTEM TO A RSTS SYSTEM

by Robert B. Goldstein and Joseph E. Duncan

Eye Research Institute of Retina Foundation,
20 Staniford St., Boston, MA 02114

Abstract

The daily automatic transfer of medical information is being done from a MUMPS-based system to a RSTS-based system. The MUMPS system is used for medical group practice management, whereas the RSTS system is used for research purposes. Medically significant information that is collected as part of the billing function is transferred from one system to the other. This is done automatically; the procedures that accomplish the transfer are built from low-cost, commercially available software. These procedures have been found to be intolerant of error and have a high failure rate.

Introduction

ERI (the Eye Research Institute) maintains a medical database consisting of ophthalmic treatments and diagnoses of patients of several groups of ophthalmologists. For several years these data were collected as part of the patient billing application that was done on ERI's PDP11/70 computer running the RSTS operating system. For various reasons, including the desire to separate research and business functions, the patient billing application was removed from the PDP11/70 and put on a PDP11/44. The PDP11/44 was purchased with software for group practice management from Interpretable Data Systems running under the ISM-11 Intersystems MUMPS operating system.

Although moving the business functions to a different computer solved many problems, it created the problem of continuing to collect the data for our research medical database. This necessitated the development of an automatic transfer capability that daily transfers demographic and medical data from the MUMPS to the RSTS system. The transfer system was built primarily from commercially available tools.

Environment

ISM-11 did not support DECNET, and no commercial tools were available that specifically connected MUMPS to RSTS. In addition, the ERI staff was unfamiliar with the MUMPS environment. Although we were very familiar with the RSTS environment, we wanted to avoid custom programs, since that would consume more resources than we were willing to commit. We therefore purchased CALOUT, from Clyde Systems, which had the capability of allowing a RSTS user to effectively become a terminal on any other system. It also allowed binary file transfer with error-checking between selected system types and ASCII file transfer between any systems with no error checking. Simplicity and the ease of implementation dictated that ASCII file transfer be done with no error-checking. The error rate has been negligible because of the shielded, twisted pair cable that connects the computers. Transmission errors would have required a change in strategy.

Other tools that were available were DEC's PIP program (Peripheral Interchange Program), DEC's ATPK program which enables execution of command files, the FORCE feature of DEC's Utility package, and the DEC Batch Processor.

The database management system used to maintain the data is ORACLE. ORACLE Corporation provides a utility ODL (Oracle Data Loader) that moves arbitrarily formatted files into the database. IDS provides a utility to transfer daily charge and registration data to tape or to an output device.

Using all of the above available tools, it was possible to construct a completely automatic process, initiated by the PDP11/70, to transfer the data. It starts itself at midnight, logs into MUMPS, and completes the transfer by about 2 AM. The captured data are then filtered, reformatted, and loaded into ORACLE.
Procedures

The procedure is shown in Figure 1.A. It is started by a system program that automatically queues nightly system jobs every midnight. Each section (A, B, C, D, E) shown in the figure is a "command file." Command files have the capability of invoking other command files. Therefore sections B, C, D, and E are invoked by A, the main command file.

The main command file sets up the I/O lines used in the transfer. Two lines are used (both directly connecting ports on the two machines with null-modem cables). One is used to receive the data and the other to log in to MUMPS to tell MUMPS to begin the transfer. These two lines are set to the correct speed (2400 baud) and cleared of any spurious signals. The main command file then invokes the other four command files as shown.

Command files B and C are similar, differing only in that one controls the transfer of demographic data, and the other controls the transfer of medical data. KB1 is set up to receive the data (DEC's PIP program is run at that keyboard). CALOUT is run on KB2 to log in to MUMPS, which is instructed to start sending the demographic (or medical) data. Because it would have been difficult to have MUMPS signal RSTS that the transfer was complete (and also because

FIGURE 1

A

Setup Lines

Capture Registration Data

<table>
<thead>
<tr>
<th>Capture Medical Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transfer Registration Data to ORACLE</td>
</tr>
<tr>
<td>Transfer Medical Data to ORACLE</td>
</tr>
<tr>
<td>Queue Logs for Printing</td>
</tr>
</tbody>
</table>

B

Set up KB1 to Receive Data (PIP) (FORCE)

Login to MUMPS Using KB2 (CALOUT)

Tell MUMPS to Send Registration Data

SLEEP for 90 Minutes

Do Closing Sequence on KB1 and KB2

C

Set up KB1 to Receive Data (PIP) (FORCE)

Login to MUMPS Using KB2 (CALOUT)

Tell MUMPS to Send Medical Data

SLEEP for 45 Minutes

Do Closing Sequence on KB1 and KB2

D

Separate New Data From Updates (SQL)

Reformat and Select Desired Data Elements (CHGIDS)

Apply Updates (SQL)

Insert New Data (ODL)

E

Reformat and Select Desired Data Elements (CHGIDS)

Insert New Data (ODL)
CALOUT does not run correctly when "detached", we have taken the approach that the controlling jobs should "sleep" for the maximum amount of time for the transfer to take place (90 and 45 minutes, respectively). After this "sleep," closing sequences are performed that consist of putting end-of-file markers on the data, terminating PIP and CALOUT, logging out the two keyboards, and saving the data in temporary, intermediate files.

Although at this point all the data have been "captured," they still must be transferred to ORACLE. In the case of the demographic data, there is no indication of whether the received data are update or new data. We distinguish between the two by checking the key field of the data (a patient ID number) against the maximum key presently in the database. Two command files are created, one to apply the updates (via SQL) and the other to load the new data (via ODL). These files are created by an in-house program, REGIDS, which also reformats the data, checks for invalid data, and eliminates unwanted fields.

Because the medical data are "time-sequence" data, all events (treatments and diagnoses) are treated as new data. All that needs to be done is to reformat the data and to select out the unwanted fields. (Some fields are business related, and only the medical information is kept.) This is done by another in-house program, CHGIDS, which creates a command file to load the data (via ODL) into ORACLE.

Results

In terms of keeping a current, duplicate record of medical events in a research database, the results have been satisfactory. We are able to use the power of ORACLE to answer complex questions using data that are, at most, 2 days old. If it were possible to perform these queries on the MUMPS machine (using "Datasearch"), it would slow down the business processing and also take more time.

However, the automatic transfer fails about 30% of the time, necessitating manual transfers the following day. The reasons for failures are: (a) one of the machines is down; (b) one of the machines is excessively busy, causing the transfer to exceed the assumed maximum; (c) paper jams, causing XOFF to be sent, stopping the transfer; (d) ORACLE "hangs" several times per day. The procedure for recovery differs, depending upon which failure has occurred. Recovery requires from 1 to 3 man-hours.

Conclusion

We have demonstrated that a low-cost automatic transfer facility between MUMPS and RSTS can be built using primarily commercially available software. However, the resulting system is intolerant of errors, inflexible, and has a high failure rate. To undertake a system like this, a significant time commitment must be made to monitor it and to provide necessary recovery procedures when failure occurs.

References