THE INCREMENTAL DEVELOPMENT OF A MUMPS-BASED CLINICAL LABORATORY INFORMATION SYSTEM

Philip Blume, Jerry Burns, and Lynn Hyde

Department of Pathology
Good Samaritan Hospital and Medical Center
Portland, Oregon 97210 U.S.A

Abstract

Modules to produce cumulative clinical pathology reports, index and report surgical pathology data, maintain a patient demographic database, and index literature references have been added to our laboratory information system. MUMPS facilitates the modular incremental approach used to develop these applications. Each module is created, evaluated, and modified until logically correct and acceptable to the user. Our library of programs is being adapted to establish systems in medical records, biomedical physics, radiology, and the pharmacy.

Introduction

In 1980, at this symposium,* we described our approach to the development of a clinical laboratory information system. By that time we had dealt with many aspects regarding the foundation of the system. The principal feature was the comprehensive test directory which contains test-specific information necessary for the system to execute functions which existed then and those which could be anticipated. For ease of development we began by producing a number of peripheral functions, such as those which process billing information and those which compile workload statistics. The development of these functions and manager-level support functions provides a base upon which all future functions may stand.

Perhaps the most significant feature of our system is the "modular incremental approach" we have taken. We isolate discrete problems and solve them in a manner which, while relatively circumscribed, retains design features which allow all of the modules to form a large integrated package. Typically, but not necessarily, these modules involve computer-based functions. We have chosen not to use any of the commercially available systems we have seen, largely because we feel that an automated reporting system must function at least as well as our manual one. We have, instead, created a module which closely emulates our manual system and actually improves upon it.

Modular development is significant also in that it may be accomplished incrementally. New software is developed and tested on existing equipment. Hardware is purchased only when the programs are logically correct and acceptable to the user. Functional modules are developed and tested, and accepted or rejected with minimal risk of capital loss and instability of the system. This approach may be viewed in contradistinction to the common employment of large, expensive and functionally monolithic systems which by their size, complexity, cost and attempted initial scope, tend to dominate the organization in which they are located. Despite limited flexibility and susceptibility to modification, in the event of incompatibilities between such systems and the needs of the users whom the system is intended to serve, there is likely to be a tendency to force compliance with demands and constraints of the large system on economic grounds alone.

MUMPS is significant in facilitating this modular approach to system development. We are using Digital Equipment Corporation's DSM-11 running on PDP 11/34 (copyright) processors. Because MUMPS affords great ease in the creation and modification of major disk databases, it is not necessary to anticipate initially the total design of every module which eventually will constitute the total system package. One is free to design a database structure for an application, knowing that data fields may be added or that current file structure may be changed with minimal effort. (System performance is affected by the structure of the global files.) While a thorough analysis of any system is certainly desirable, it is not possible to anticipate every subtlety of an implementation or every user requirement. With MUMPS, we create new application programs with extreme rapidity and constantly hone them. Creation of the system as a whole consists of writing new code, evaluating it in a real environment, modifying it, and repeating the sequence. By the time the programs are completed they have been evaluated. Assuming satisfactory response time with increasing use of the system, their success is assured. As our library of general-purpose subroutines grows, each new major function becomes easier to develop because much of the logic and code can be drawn from previous work.

Since our earlier report we have developed a variety of additional modules, including a literature reference indexing system which we use for research references. This indexing system has been adopted by the pharmacy for handling their drug information

* Proceedings, Fourth Annual Symposium on Computer Applications in Medical Care, Washington, D.C., 2-5 November 1980, Volume 1, p. 495.

0195-4210/82/0000/0297$00.75

file. We have produced a surgical pathology reporting and indexing system. We are presently involved in the creation of a clinical pathology cumulative reporting system which will be the most significant feature of the package until we can use the computer to aid directly in diagnosis.

Literature Reference Indexing

The literature reference index module, or the BIBLIOGRAPHY function, allows the storage, selective retrieval and display of literature references in standard journal citation format. In addition, it handles a wide variety of other information sources such as books and slides. Each item is assigned an identification number in sequence. By means of a reference entry dialog, the user creates a record in the computer containing the necessary information about that reference.

A numeric code is assigned each journal, author name and keyword established in a user's file. The system stores only the appropriate numeric codes as a part of each reference record. A directory which is part of the single global array containing the user's entire file lists the name of the journal, author, and keyword assigned to each numeric code. The keyword file has the format:

```
^BIBFILE("K",0) = number of keywords in file
^BIBFILE("K",n) = keyword n
^BIBFILE("K",keyword n) = n
```

The author index, ^BIBFILE("A",...), has the same structure. The journal index accommodates both the full name of a journal and an abbreviation and has the form:

```
^BIBFILE("J",0) = number of journals in file
^BIBFILE("J",n) = full name of journal n| abbrevi-
                  ated name of journal n|
^BIBFILE("J",full name of journal n) = n
^BIBFILE("J",abbreviated name of journal n) = n
```

The information for each reference is stored as several different records so that searches are reasonably efficient. The records for a given reference have the form:

```
^BIBFILE("REFERENCE",n) = journal number|volume|
                          year|first page|last
                          page|
^BIBFILE("TITLE",n) = title of reference n
^BIBFILE("KEYWORDS",n) = |keyword code 1|code 2|...
^BIBFILE("AUTHORS",n) = |author code 1|code 2|...
```

The file may be searched to find all references with any combination of keywords and authors by using the '$N' function. Similarly, the titles on file may be searched for a specified string of characters. Searches are not particularly fast, but as we use the system, speed is not an issue. We have elected not to expend additional disk space to create pointer files.

An option exists which allows the specification of a series of footnote designations and associated file reference numbers. A bibliography can be printed in a format suitable for attachment to a manuscript. A similar option exists which allows permanent storage of a list of the user's own publications so that at any time a bibliography may be produced for a curriculum vitae.

While the BIBLIOGRAPHY function was created for the individual use of departmental staff members, its creation had some broader ramifications. The pharmacy department has a drug information division to respond to queries from physicians about pharmaceutical administration. Each time a particular question is studied, a file is created for use in responding to similar future queries. The BIBLIOGRAPHY function handled this file without modification. Additionally, the file structure and search logic used in the BIBLIOGRAPHY function were adapted to accomplish the coding of sites and diagnoses for the surgical pathology module.

The Patient Demographic Database

The patient demographic database is central to all functions of the program package which deal with patient-oriented information. The medical records department has a master index file containing about 300,000 patient references. Each area applications package interacts with this central file.

Apart from secondary index and pointer files, all clinical information is filed according to the patient's medical record number, 'RNO', presently a six-digit number with an additional check digit. The local demographic database consists of two globals. ^DN(), which sorts information by last name, first name, and 'RNO', has the structure:

```
^DN(LN,FN,RNO) = nursing station
^DN(LN,FN,RNO,label) = data
```

'Label' identifies such data elements as date of birth, account number, height, weight, etc. ^DR() is sorted by 'RNO' and serves as a pointer file to ^DN(). It has the structure:

```
^DR(RNO,"FN") = first name
^DR(RNO,"LN") = last name
^DR(RNO,"NS") = nursing station
```

There are three functions which deal with the local patient demographic database and are called as options from the main menu. The ADMIT option permits the entry of new patients to the database. It also permits modification of the demographic data and all files dependent upon that data. The DEMOGRAPHIC DISPLAY option permits a query of the database to display all demographic information associated with a patient. The DISCHARGE function removes demographic information from the local database and purges unnecessary information. It also interacts with the master patient file to assure the preservation of information which must be stored permanently.

The ADMIT (and edit) function is handled by a program which, as indicated above, may be called as a main menu option. When so called, a switch variable is set to provide for the entry of multiple patient records. This program may be called also as a subroutine during execution of any program which allows the entry of information for a patient who is

not included in the demographic database. In this case an alternate setting of the switch variable is used and effects a return to the calling program after the entry of demographic data for a single patient. (We try to employ this dual-function technique wherever possible.)

Most application programs interact with the demographic database via a single subroutine which requests entry of 'LN', 'FN' and 'RNO'. The subroutine then determines if data for the patient exists. A valid 'RNO' (with check digit) may be entered at any point to identify the patient. A number, such as a surgical pathology accession number (with check digit) which is uniquely associated with a medical record number, may be entered in selected instances to identify the patient.

The system uses a series of default medical record numbers which have an additional, nonnumeric, leading character and are automatically assigned by the system. These numbers do not repeat. They are used to file information before a formal record number is issued. This technique permits the filing of laboratory data for specimens submitted on patients not seen in our hospital. Without this important feature we would have to bear the expense of assigning and maintaining regular medical record numbers for all these specimens, exclude such data from our file, or use two separate and less suitable file structures.

An index function which displays all names in the file starting at any specified point in the alphabet facilitates use of the demographic file. A wildcard search function permits a single letter of a last name to be replaced by an asterisk. This function provides a display of all records with a last name matching the specified name, but with any allowed character in the position occupied by the '*' (e.g., 'PETERS*N' displays all records on file for 'PETERSEN' and 'PETERSON').

Surgical Pathology Function

The surgical pathology function was intended originally to serve as an index to typewritten surgical pathology reports and to provide access to cases with specified diagnoses. We decided against using standard nomenclature (e.g., SNOMED) because the difficulty encountered in getting staff members to do the coding jeopardized the entire project. We adopted the same index structure used in the BIBLIOGRAPHY function. Although we risk having more than one code for a particular diagnosis, we review the index before a search and request all pertinent codes. This approach lacks the elegance of more rigorous coding procedures, but its simplicity serves the purposes of our department.

We shall not describe the logic of the SURGICAL PATHOLOGY function in great detail. One feature is of particular interest. We created a file structure which permits a text of unlimited length to be entered in lines of up to 80 characters. The general file structure for such a data component is:

```
^S(RNO,accession number,label) = number of lines n
^S(RNO,accession number,label,1) = line 1
^S(RNO,accession number,label,n) = line n
```

This structure was adopted to provide maximum flexibility in the storage of data, but it also permits the report to be filed in its entirety. Although this function originally was intended to serve only as an indexing system, we now use it for the actual production of all surgical pathology reports. Figure 1 is an example of such a report. A text editor which functions as a simple word processor is included. The printing of multiple copies of reports eliminates photocopying costs. The full reports will be kept in the system as long as practical and then will be transferred to magnetic tape for microfiche production. The index data will be stored permanently. Since report generation is by direct terminal input rather than typing, the index function involves little additional labor.

Cumulative Clinical Pathology Reporting

The laboratory information system must be capable of handling a wide variety of functions related to order entry, specimen collection, data acquisition and verification, and reporting. We are in the process of developing functions to deal with all of these areas. One of the most important functions performed by the system is the production of laboratory reports. These reports should be presented in a cumulative fashion. They must be easy to read and use. Logical design and esthetic appeal affect readability. Many computer-based clinical pathology reporting systems fail in this respect. In an attempt to produce a cumulative report each time new laboratory work is ordered, they generally compress data in a form which is difficult to read.

The manual system in use at this institution and used for several years at the Univeristy of Minnesota is logical and efficient. It produces reports which represent the benchmark with which all other reports must be compared. It was used by others before we adopted it. David Seligson used such a report format at Yale University. We borrowed freely from its design in producing ours. John B. Henry* described the approach as well.

It has been our goal to develop a computerized reporting module for our laboratory information system which is at least as good as the manual system it replaces. The computerized module uses a logical design which, while not in use in all laboratory departments as this is written, promises to be capable of handling any future contingency. With minor changes in layout, we produce reports which are virtually identical to and actually better than our manual reports.

All necessary procedural information about individual tests is obtained from the test directory. All necessary patient demographic information is obtained from the demographic file. For the purposes of order entry and reporting, the primary patient identification is the medical record number, 'RNO'. Specimen accession numbers, always tied to the 'RNO', are used extensively.

---

* Todd-Sanford, Clinical Diagnosis by Laboratory Methods, 14th Edition, I. Davidson and J.B. Henry, editors, W.B. Sauders, 1969, p. 495.

Figure 1. Example of computer-generated surgical pathology report.

Good Samaritan Hospital & Medical Center
PORTLAND, OREGON 97210 DEPARTMENT OF PATHOLOGY

Date: 18-JUN-82 Path no: S82 04878-6

KASEBERG,
NAME

148560-6 F 22-OCT-1909 6C 639
RECORD NUMBER SEX DATE OF BIRTH PHYSICIAN STATION ROOM-BED

PHYSICIANS
1. W Parsons - S-370
2. Histology
3. AND

SPECIMEN: I TISSUE FROM RT. FRONTAL LESION
II TISSUE FROM RT. FRONTAL LESION
III BONEY ATTACHMENT OF MENINGIOMA

GROSS: I. Present is a pale pink fragment of tissue measuring 1.0 x 1.0 x 0.5 cm. This specimen is submitted in toto, 1 cass.
II. Present is a single pale ivory to pale yellow fragment of tissue measuring 2.0 x 1.2 x 0.3 cm. The frozen section material is submitted in cass. FS. Remainder of tissue is submitted in toto, cass. A.

FROZEN SECTION DIAGNOSIS: Tumor, probably meningioma. (AND)

III. Present is a single irregular fragment of pale pink to ivory-white cortical bone measuring 1.7 x 2.0 x 1.5 cm. Several sections are cut and allowed to fix before decalcification.
K Gunson/AND 18-JUN-82

MICROSCOPIC: A meningothelial meningioma is infiltrating fibrofatty tissue. Nuclei are bland and there is no mitotic activity.
A portion of the skull has been submitted for decalcification.
AND 19-JUN-82

DIAGNOSIS: Brain, right frontal, tumor: Meningioma.

A. N. M. D.
21-JUN-82

DEPARTMENT OF PATHOLOGY SURGICAL PATHOLOGY PAGE 1

Figure 2. Example of computer-generated chemistry report.

Good Samaritan Hospital & Medical Center
PORTLAND, OREGON 97210 DEPARTMENT OF PATHOLOGY

ADAMS,
NAME

999999-6 F 01-JAN-1900 G HEALTH4C 457
RECORD NUMBER SEX DATE OF BIRTH PHYSICIAN STATION ROOM-BED

TEST		REFERENCE RANGE	LAB NO.->	01080	01079	01081	01083	01084	01085	01087	01088		
			DATE 1982	05/22	05/22	05/22	05/22	05/22	01/08	05/26	05/27		
			COL'N TIME	00:25	00:25	00:45	06:30	09:15	19:30	07:00	07:45		
CALCIUM	S	8.6 - 10.6	mg/dl			10.0							
PHOSPHORUS	S	2.2 - 4.5	mg/dl										
MAGNESIUM	S	1.4 - 2.4	mEq/L										
IRON	S	65 - 175	ug/dl										
TIBC	S	250 - 410	ug/ml										
UREA NITROGEN	S	7 - 20	mg/dl			20	18				19		
SODIUM	S	137 - 140	mmol/L			148	144			142	143		
POTASSIUM	S	3.7 - 4.9	mmol/L			4.3	3.8			4.0			
CHLORIDE	S	95 - 106	mmol/L			110	106						
BICARBONATE	S	22 - 30	mmol/L		22	23	27						
GLUCOSE	S	59 - 107	mg/dl										
CREATININE	S	0.6 - 1.3	mg/dl										
PH	B	7.35 - 7.45											
PCO2	B	38 - 48											
PO2	B	[illegible]											
O2 SATURATION		[illegible]	mg/dl										
[illegible]	S	3.8 - 4.5	g/dl			4.0							
[illegible] PROTEIN	S	6.4 - 8.1	g/dl			7.1							
CHOLESTEROL	S	150 - 250	mg/dl			223							
TRIGLYCERIDE	S	30 - 135	mg/dl										
URIC ACID	S	2.9 - 5.7	mg/dl										
OSMOLALITY	S	275 - 300	mOsm/kg										

DEPARTMENT OF PATHOLOGY CHEMISTRY I PAGE 1

The global database may be conceptualized as a file of matrix cards arranged in alphabetic order by patient name. (Note that MUMPS is well suited to this approach since it can deal easily with largely empty matrices without significant degradation in storage efficiency or increase in program complexity.) The data file is organized by nursing station, room/bed number and medical record number, so that reports may be printed in distribution and charting sequence. The fourth subscript of the data file is the REPORT FORMAT NUMBER.

The system permits the user to create an unlimited number of different report formats falling within a limited number of hard-coded types called GENERIC FORMAT TYPES. For example, the columnar report illustrated in Figure 2 containing a maximum of 37 tests and 11 five-character results columns employs 'GENERIC TYPE 0'. Most of chemistry and hematology data will be reported on forms of this type. As this is written, type 0 is the only type we have created and tested. However, the logical design provides for a form with 41 rows by one 110-character column defining a blank, horizontally held page and thereby permitting the display of free-form text-- the extreme form of nonstructured report. Other report formats will fall between these extremes and will be handled easily by the logic of this system.

The implementation of this simple approach involves the creation of three different series of subroutines. A single subroutine exists within each series for each generic report type. Although the creation of a new generic report type does not require significant modification of existing programs, it does involve programmer intervention. By adding one subroutine to each of the three series, the system can accommodate a new generic type. The user may create any number of report formats employing existing generic types.

Report Format Creation. Manager-level functions permit the creation of new report formats. Following specification of the generic type upon which the format is based, control is transferred to a subroutine which is named for that type. This subroutine asks the user to enter the specific information necessary to create a report of that generic type. For example, in the case of generic type '0', it would request entry of the number of tests, test names, footnote texts,etc., assign a unique format number, and establish the record in ^RFM(), the format global. This record associates each format number with its generic type. The information that would be stored for a format of generic type '0'has the form:

```
^RFM(format number) = format name (e.g., 'CHEMIS-
                      TRY-1')
^RFM(format name) = format number (e.g., 5)
^RFM(format number,0,0) = generic report type
^RFM(format number,0,1) = number of data rows
^RFM(format number,row,1) = name of test
^RFM(format number,row,2,"F") = female reference
                                range
^RFM(format number,row,2,"M") = male reference
                                range
^RFM(format number,row,3,"F") = female low flag
^RFM(format number,row,3,"M") = male low flag
^RFM(format number,row,4,"F") = female high flag
^RFM(format number,row,4,"M") = male high flag
^RFM(format number,row,5) = units of measure
```

In addition to information dealing with specific tests, other information may be stored. For example, if ^RFM(format number,row,1) equals "-", the seven elements pertaining to a test are absent. The system prints a horizontal dividing line on the report at that location. In a similar manner rows may be reserved for use in displaying footnote codes, technologist identification, etc.

Result Storage. The test directory listing for each procedure now includes values for the REPORT FORM on which the result is to be displayed and for the REPORT FORM LOCATION which specifically places the result on that form. When a particular test result is placed in the database, this information is available. By means of the format global, ^RFM(), the value of REPORT FORM allows the generic format type to be identified. Control is transferred to one of the second series of special subroutines which creates the appropriate type of database records for the given generic type.

Result Display. When reports are printed, the printing program passes through the results database global, ^D(). Within the section containing data for a single patient, the information is stored by REPORT FORMAT NUMBER. By use of the format global, ^RFM(), the generic report type is determined. To produce the report document, control is passed to one of the third series of special subroutines which uses both the data in the results database, ^D(), and the report format database, ^RFM().

Only those pages which report newly ordered data are printed. These pages are printed during each report generation run, only until a report has been produced for which results are no longer pending. At this time the card is returned to inactive status. When a card is filled, it must be marked 'PERMANENT RECORD--RETAIN IN CHART' and not used again. Within the result database, elements called 'PRINTFLAG' and 'FULLFLAG' exist for each defined page. Appropriate logic exists to set these flags to achieve the desired functional characteristics.

The results database global, ^D(), is organized by nursing station, room/bed number and medical record number. Within the database section for a particular patient, the next division of the data is report format number, 'RFNUM'. This number uniquely identifies the generic report type. ^D() contains data elements which deal with the report page as a whole and are independent of generic report type. Additionally, ^D() contains data elements which are dependent upon the generic report type.

Conclusion

The modular incremental approach is well suited to the creation of a laboratory information system. It is also well suited to other areas of the hospital. We are well into the installation of major applications in radiology, medical records, biomedical physics, and the pharmacy. We hope to make these projects the subject of future reports.