AN INTEGRATED COMPUTER-BASED BIBLIOGRAPHIC DATA SYSTEM FOR A LARGE UNIVERSITY LIBRARY: PROBLEMS AND PROGRESS AT THE UNIVERSITY OF CHICAGO

Charles T. Payne

The University of Chicago Library is working on a project for the development of an integrated, computer-based, bibliographical data system for a large university library. This project title is awkward, but it is descriptive, and it stresses some important features of the project: (1) a third generation computer with random-access disk files and remote terminal access capability is being used to implement a real-time library data processing system; (2) the library system design is based on the operational requirements of a large university library and does not attempt to solve the problems of other types of libraries nor those of national systems; (3) the system is highly integrated in that all processing data for the technical processing operations (acquisitions, cataloging, binding, etc.), as well as bibliographic descriptive data, are handled in a single system; and (4) the title emphasizes, we hope, that this work is developmental and experimental.

The long range goals of this project are those of a totally automated library system and thus can now be defined only in general terms. Enormous problems will have to be faced before any sort of total library system can be achieved, including problems such as file conversion, massive file organization and management, and related searching techniques. Even in the early stages of development, where the system design has been defined in detail, much original development work had to be done in implementing and, at times, in nudging forward the state of the art of computer applications. Nothing in our development to date has shown our approach to library automation to be invalid. It will be some time, however, before we have proceeded far enough and have had sufficient operating experience to make meaningful evaluations of costs, benefits, and performance.

A number of factors have helped shape the project design and development. Among the most important is Herman Fussler, the

Charles T. Payne is Library Systems Analyst, University of Chicago Library.

Director of the University of Chicago Library, who has long been an advocate of the application of machine technologies to solve library problems and who has expended a great deal of effort on the problems of how libraries, both locally and at national levels, can best take advantage of the emerging computer technology. His influence has been important in the development of this project both in long-range planning and in handling of immediate realities. A second factor is the situation at Chicago which demands consideration of both the immediate problems of a badly overcrowded Harper Library and the expectation of vastly different operational conditions in the projected new Joseph Regenstein Library. This situation has affected both the project schedules and the system design. The third major factor influencing the project development was the emergence of a third generation computer with its enormous potential for library systems development. This came during our planning phase, and we made the decision to by-pass punched card and batch processing development and move directly to an on-line operation. The project has cut its teeth on-line, so to speak, and it has been a learning experience. Another and most important factor in the development of this project has been the generous cooperation of the National Science Foundation in providing a substantial portion of the funding. Progress on the project would have been much slower without this aid.

It is good to have long-range plans, even if ill-defined, but in the real world of system design and implementation one must proceed in well-ordered stages and must work within the confines of the possible, or almost possible, and make use of on-the-shelf equipment. The first stage in the project was called the Book Processing System. The remainder of this paper will be a description of the Book Processing System, its implementation, problems, and present state, followed by a discussion of the implications of on-line technology.

In library technical processing operations almost all work can be described in terms of data handling, including those intellectual operations in book selection and in cataloging. The over-all design feature of the Book Processing System is to incorporate all data relating to an item being processed into a single machine record. This includes the bibliographical descriptions, the various processing information such as dealer, fund, bindery, etc., and other operational data such as time in cataloging, cataloger's identification, etc. All production output is generated from the machine records, including orders, claims, cancellation or confirmation notices, fund accounting and invoice processing reports, catalog cards, bindery tickets, book pocket and spine labels, distribution lists, processing file lists, operational statistics, etc.

The potential savings by elimination of repetitious copying, sorting and file maintenance are substantial. In addition, from a given point, the library is creating machine readable records for all

materials processed. This seems to solve the problem of machine readable records for future material. If only someone would now solve the problem of the past—the problem of conversion of retrospective records.

In order to handle all of the different data by machine it is necessary to identify and define each element of information that goes into a record. These elements of data are identified by tagging codes. The data elements (with few exceptions) and therefore the item records, are variable in length. Basically, our tagging codes are three-digit numbers which define the various elements of information, e.g., 010 item number, 035 dealer, 520 title, etc. In addition, certain tagging codes initiate actions within the computer system at the time of input. Other codes can change the status of a record within the file. Still others affect or direct output processing and formatting. Every tagging code has a definition that includes input requirements, data content and form, and (if any) resulting processing operations. A numerical code scheme for machine identification of data elements was adopted early in the development of this project. Tagging codes had been used elsewhere for handling bibliographical data and no obviously superior scheme was readily apparent.

Counts of the total number of possible elements in bibliographic and processing data quickly led to use of a 3-digit number. This provided enough code positions so that the various types of entries were assigned individual codes. The output format, therefore, can be defined by the tagging code and no references need to be made elsewhere to determine this. It was possible to build in relationships between the various types of entries.

Over one hundred specific tagging codes were assigned, about two-thirds of which cover bibliographic description. The various elements that make up the complete record can be input at different times. The first input creates the record in the machine file. In operation, the elements of data are input at the time of generation in the library, e.g., at the time of ordering and receiving material, or when it is cataloged. We also have editing routines for correcting or altering previously input data.

The computer used for implementation of this system is an IBM 360 model 30 computer (see Figure 1). This is a small computer and it has a modest core size of 32 K bytes. The computer and peripheral equipment are housed across campus from the Library and its use and costs are shared with other groups. Simultaneous, shared use of the computer has been worked out for applications of both the Library and the Maniac III experimental computer project of the Institute for Computer Research. This is not, however, a large scale, time-shared operation in the true sense.

Connected to the computer are two disk drives with about 14.5 million character on-line storage capacity. The Library has two IBM

1050 terminals. These are connected by telephone line to the computer. Each 1050 unit consists of a typewriter keyboard and printer, paper-tape reader and punch, and an auxiliary printer. The printing elements are the easily interchangeable "golf ball" type, similar to the Selectric typewriter, and offer the potential of providing libraries with a very large character set capability, at a relatively modest cost. It will take some sales potential, however, to persuade the manufacturer to develop the character set potential.

The type of terminal described above is known as a slow speed terminal. The maximum line transmission rate is about fifteen characters per second—much slower than a high speed line printer. We are using the 1050 for printing catalog cards at the present time. The printing speed is slow, but this is not a disaster in terms of computer time when operating in a shared, or multi-programming mode. It takes the computer the same amount of time to format a card for output whether for a high or slow speed printer, and it is free to do other work while printout is going on.

We do not see the 1050 as a permanent solution for the bulk of production printout. We will probably continue to use these or similar terminals for input, for production printout needed in the library on a tight schedule, such as orders, foreign alphabet printout material, for worksheet printout, etc. For the bulk of catalog card printout, a high speed line printer with the universal character set may be used if it becomes available. Various reported developments in non-impact printers sound very interesting and may be a future possibility. In any case, efforts have been made to keep the machine records independent of the means of input or of output. The stored machine records are coded in 360 internal code. The records as such are not formatted for output, but are merely strings of tagged data. The machine record does not look like a catalog card. This data handling system should be versatile enough to allow for any future developments in input-output equipment.

So far discussion of the Book Processing System has concerned data handling—the item record and its elements—and equipment; this was preliminary to talking about programming. In any computer processing, there are two aspects of the programming to consider—the system software, or computer operating system programs, and the applications programs. In the normal course of batch processing, the computer system software can be pretty much ignored. The programmer adapts to the existing operating systems and languages in the development of his programs. Further, most university computation centers have developed large, established sets of system software that will cover almost any need.

It is when the basic operating software for a computer system does not exist that it becomes most noticeable. Our early experience on the 360/30 was with an "undebugged" basic operating system—data

file management programs that no one could make work, autotest programs for "debugging" that were themselves "undebugged," and a complete absence of teleprocessing control programs. In effect, although we have called Book Processing implementation our first phase, our actual first phase was to develop an operating computer system.

The situation is vastly improved now. New and improved software packages have been released by the manufacturer. It is not even absolutely necessary to program in assembly language anymore. But the basic lessons are still there: (1) the software must perform the functions required; (2) it must be thoroughly tested; and (3) in any on-line operation it must be resident in core at all times. This latter can become a critical factor in simultaneous shared use of the computer where core size is small.

The applications programs for an on-line operation must be available when called. This means that the applications programs must be resident in core or be available in on-line, random access storage. It quickly became apparent that all of the Book Processing applications programs could not be held in core. In fact, as it worked out, not even the entire set of programs for certain single operations (i.e., catalog card format) could be held at one time and, therefore, program overlays have since been used. The following pages give an outline of the programs and sizes for the data input phase (see Figure 2) and for the catalog card output phase (see Figure 3). The diagrams show the overlay arrangements that have been worked out.

For the input phase the system will:

(a) call in the proper programs on command from the library terminals,

(b) set the conditions to accept data from the library,

(c) check each incoming record against the file to see if it is a new record or update of existing record and call in this indicated data file management program,

(d) scan the input for logical errors,

(e) edit out unwanted blanks, carriage returns, line feeds, tabs, etc.,

(f) convert codes BCD to hexadecimal,

(g) scan for output distribution requests,

(h) create entry in key table for requested output,

(i) enter data into the file, and

(j) perform necessary editing.

These programs are operational and will handle input of all record elements that have been defined so far. We also have operational a major off-line set of programs that construct output stacks working from the input key table. The stack table programs work now for catalog cards only.

From the initial input for distribution, the input phase programs create a key table. The off-line stack program sorts these by type of output (now catalog cards only). From the key table and from the data files, these programs create entries in the stack tables for every card wanted. The new stack entries are merged with previous entries remaining from incomplete output printing. These stacks are sorted by location, by type of array, and by entry (or call number).

The catalog card output phase is a set of programs for on-line production output and is also operational and in use. This is a complex set of programs, but very quickly explained. It does the following:

(a) accepts command from the library terminal and calls in programs,
(b) selects stack tables for requested locations,
(c) selects card for output,
(d) formats call numbers and saves,
(e) formats text (lines and words) and saves,
(f) formats card, and
(g) prints out.

All but the print-out takes but a fraction of a second and then the computer waits to do the next card. In simultaneous operation with Maniac applications, it works for them during this period. This output gives cards for each requested location in filing array. Work is in progress on programs to include charge card, pocket label, and order printing production.

Two additional useful programs are in operation. One prints, on demand, the exact form of the machine record for any item, the only change being translation of machine code to output printing characters. The other program gives counts of the card stacks and is used to schedule output printing. The next major applications programming effort will go into fund bookkeeping, invoice handling, and other acquisitions processing.

The procedures described above are still a long way from a total functioning library system. Much work that had not been tried before has been performed, however, and some things are beginning to be known about this mode of operations. A year ago no one knew about some of the things which now appear obvious. We can see many implications of on-line processing. Some, in conclusion, are discussed below:

(1) System development will become open-ended; the almost unlimited opportunity for change and improvement is irresistible. There is no point where you can say that this is as far as you want to go.

(2) This means that the library systems development staff, the computer systems staff, and the programming staff will become a permanent part of the library and will not be just a temporary bother.

(3) The combination of high core requirements and low computer utilization indicates that future economies will be in the large, university-wide, time-shared computer rather than the computer dedicated to library use alone.

(4) On-line library operations will, of necessity, be simultaneous shared computer operations, even on a dedicated machine. By the time one implements systems for technical processing, with heavy input/output requirements, plus circulation, with immediate response needs, the library will be time-sharing with itself.

(5) In discussions of on-line operation, one frequently hears mentioned the prospect, usually imminent, of putting a terminal on every cataloger's desk or in faculty offices, or scattered around elsewhere. Our experience is that lines are expensive, the terminal equipment is expensive, and the transmission control unit at the computer end is expensive. Rather than spread terminals around we have tried to effect maximum line utilization for the terminals we have. The economics on this should improve, but until they do, it seems likely that most on-line terminals will be stationed in data processing rooms and at specific heavy-load work counters.

(6) The library needs staff members trained both in library operations and in computer science. It has been our experience that computer people will underestimate the library requirements by at least an order of magnitude even when the requirements are documented. The library needs a voice that can talk to the computer policy committee and look after the long-range interests of the library. The library should participate in the planning of computer facilities and see that its future plans and requirements are taken into consideration.

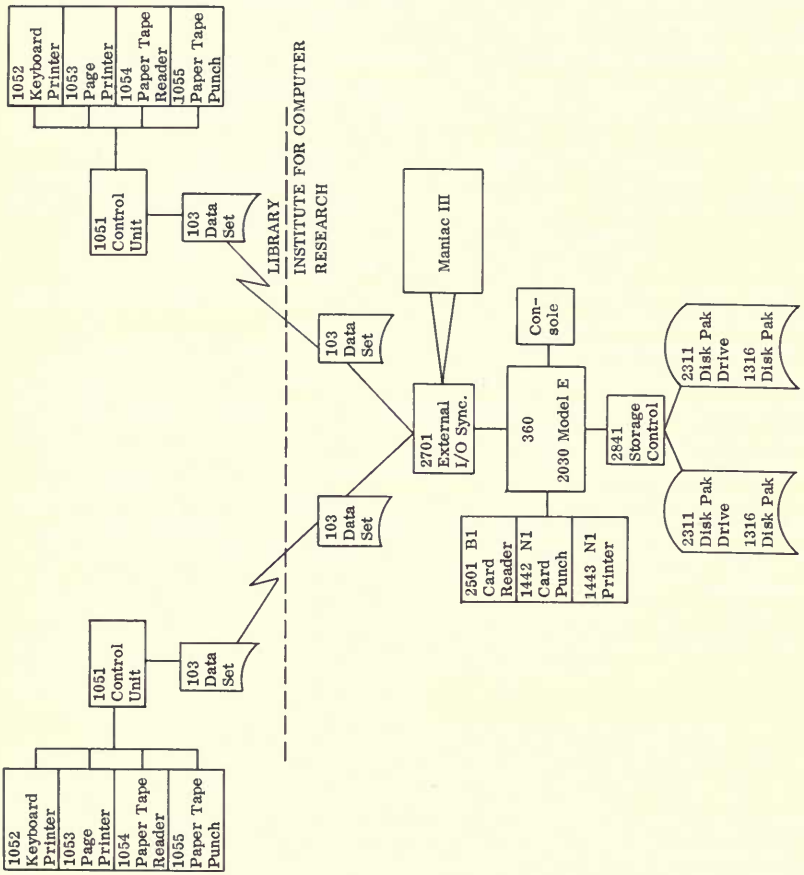

Figure 1.
Computer system and library terminals used during first year development.
On-line disc storage capacity of 14.5 million character.

INPUT PHASE		SIZE (decimal)
1. Supervisor: BOS supervisor 1050 teleprocessing control program Maniac III Input/Output control program		13,024
2. Index Sequential Data File Management	LOAD	2,244
3. Index Sequential Data File Management	ADD/RETRIEVE	3,996
4. Input Scanning Program	CONDNS	3,587
5. Paper Tape Control Module Read paper tape control program Input phase program check Initial edit of input program	 RPT PCHECK REMOVE	4,071
6. Processing Information File Update Module Logical error routine Hex to bcd conversion routing Table search routine Bcd to hex conversion routine Update to Processing Information disk file	 ERROR MAKBCD SEARCH CODEC UPDATE	4,793
7. Output Distribution Request Module Scan and construct indicator pattern from distribution request Create entry in key table for cat. cards, charge cards, and/or orders	 DISTR WTKEY	1,837
8. Historical File Request Module Flag item for historical file	 HISTRY	495
9. 1050 Command Language and Library Supervisor		2,773
		36,820

Figure 2
Input phase

INPUT PHASE (core requirements)

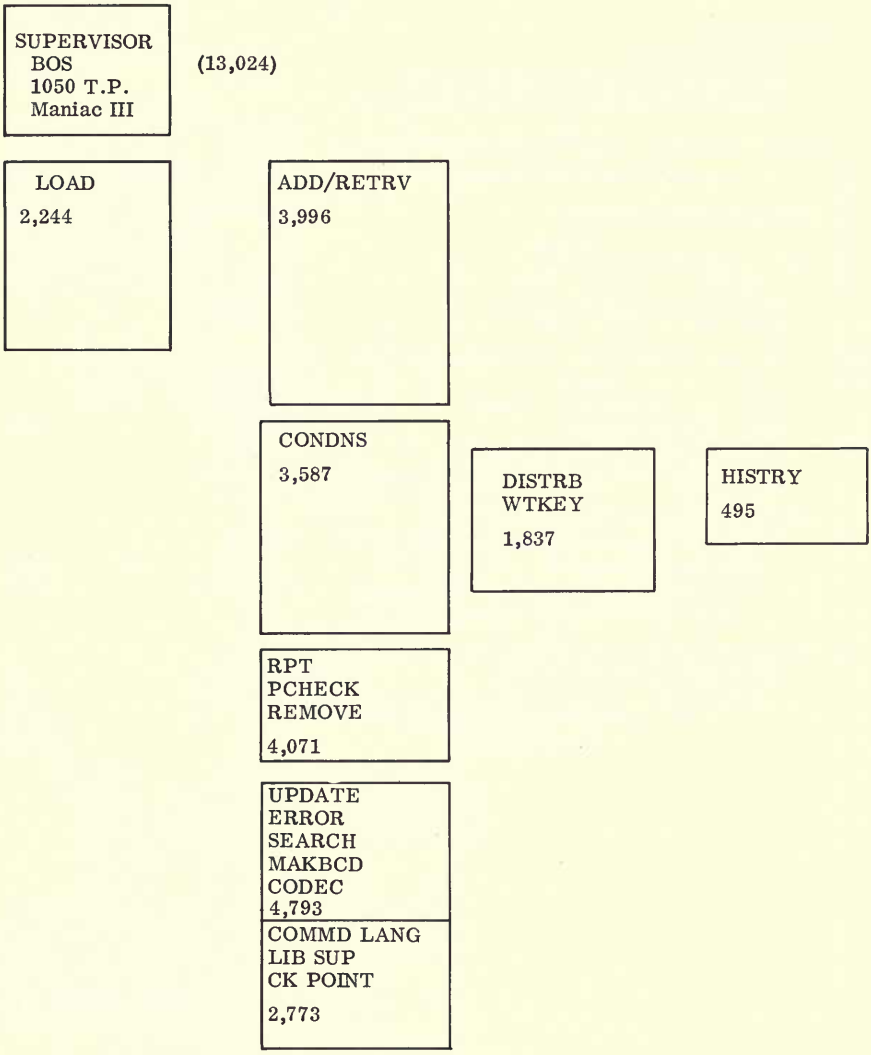

Figure 2 (cont.)

CATALOG CARD OUTPUT PHASE	SIZE (decimal)
1. Supervisor (see input phase)	13,024
2. Major catalog card controlling phase Catalog card teleprocessing supervisor Catalog card data supervisor Build item number routine Index Sequential Data Management read program Interface to I.S.D.M. for reading next tagging code Logical error routine Program check routine for catalog card phase Read stack routine (Catalog Card Descriptor records) Delay routine	13,947
3. Call number formatting routine	1,033
4. Build text of card routine Reformats words and lines	2,855
5. Build catalog card routine Merges call number, main or added entry, and text of card	905
6. Read directory to stack routine	607
7. 1050 Command language and library supervisor	2,773
Total	35,144

Figure 3
Catalog card output phase

OUTPUT PHASE (core requirements)

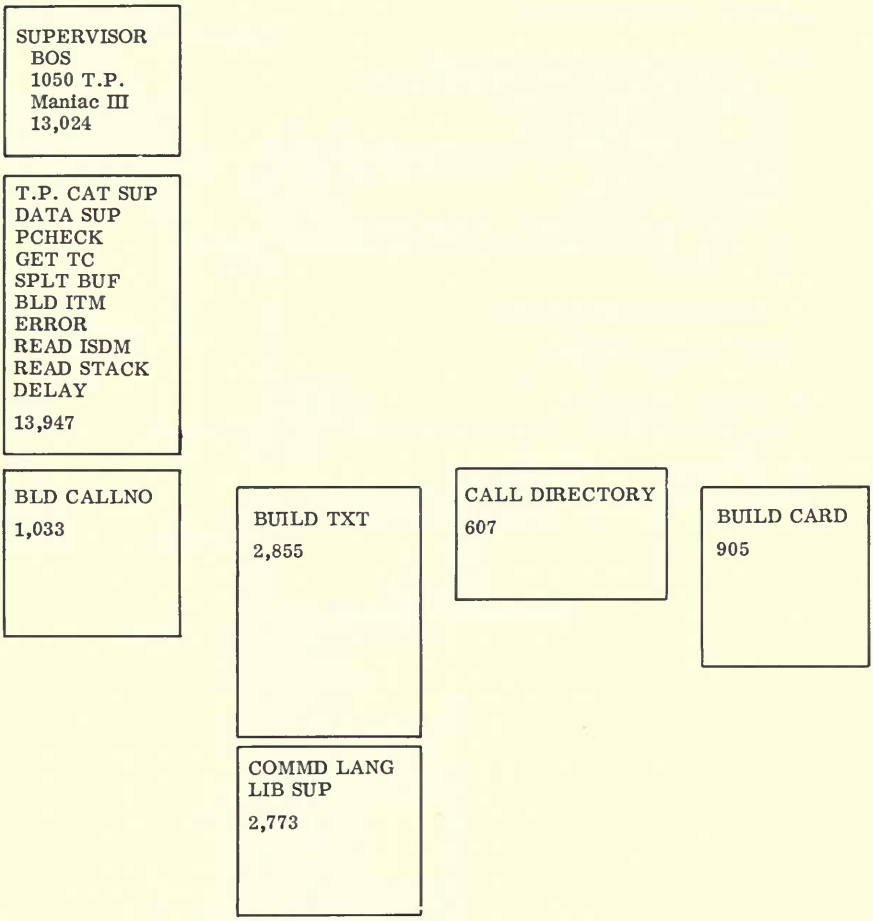

Figure 3 (cont.)