ALLEN B. VEANER

Major Decision Points in Library Automation

This article is based on a longer, more detailed paper prepared for the 1970 Midwinter Meeting of the Association of Research Libraries. Readers interested in the complete text (with bibliography) are referred to the Minutes *of the ARL meeting. The author discusses automation in the context of the management, facilities, and system requirements for large research libraries.*

Introduction

The main subject of this paper is change. Fundamental changes accompany the automation of library functions. Whether one employs batch operations, on-line techniques, or a mixture of the two, it constitutes a totally new way of life. When applied to a large central library, automation creates the most radical changes in library operations since the creation of libraries. This paper will not deal with single-application, small-scale automation efforts, nor with those in branch or special libraries. Rather it is addressed to the factors and decision points in developing a major program of automation in the main research facility of a large university or research organization.

The early questions in deciding upon an automation program are concerned with the implications of radical change. What is the status of the current manual system? Is the time right for a change? What are the known or anticipated effects of major change upon the staff? Upon the faculty and students? What are the financial implications? Answers to these questions must be as detailed as possible and must be based on realistic expectations concerning the functions which are currently susceptible to automation.

What kinds of library activity can be automated in the present state of the art? Although much research and experimentation has been conducted on advanced systems of information storage and retrieval, it is clear that there is nothing yet on the horizon to rival the human brain and natural language for intellectual tasks of great complexity. Man is still the principal thinking creature, the one who can handle ill-structured problems and heuristic inquiry. But in the library he remains heavily burdened with routine tasks, or what is more accurately called *formalizable* work. We have enough experience to know that the activity currently susceptible to change through automation is this formalizable, housekeeping work. Hence the candidates for automation are of two kinds: repetitive tasks and those jobs which are deterministic and highly structured. They must involve relatively few intellectual decisions or decisions which are both repetitive and of a compara-

Dr. Veaner is Assistant Director for Bibliographic Operations, Stanford University Libraries. *The research reported in this paper was supported in part by U.S. Office of Education grant OEG-1-7-071145-4428.*

tively lower order. "Is this number bigger, equal to, or smaller than that one?" "Is this date earlier or later than another?" Formalizable activities are also the hardest to change because habit and custom govern their performance. Thus, immediately upon embarking on a study of automation, one enters a political thicket; at issue are performance norms, standardization, organizational structure and reporting patterns, job analysis, time and motion studies, reassignments, retraining, and the upsetting of all former social and occupational stability. In summary, the question is: How well is an organization prepared for change? Is it willing to employ staff whose mission is challenge, evaluation, and program change?

The methodology of introducing change has itself changed radically. Gone are the days of changing procedures by administrative memo or unilateral fiat. Intra-institutional competition for funds has become public knowledge and is forcing the revelation of administrative and economic realities which might previously have remained behind the scenes.

Ideally, a new system of doing anything should sell itself because those directly affected by the change have already fully participated in its development. This makes good sense because no level of staff will be unaffected by an automation program. Nevertheless, much work remains to alleviate the prevalent anxiety of job loss or takeover by machines. Jacques Barzun stated not long ago that "mechanical work is the computer's meat; as a source of *intelligence* it is a total loss."[1] In comparing the human brain and the computer, Orlicky points out:

> When we discover areas of mental work in which we can outperform computers, we tend to regard computers as sluggish or clumsy, but perhaps a more proper way of looking at it would be to realize that the particular task is extremely difficult and that our own ability in this respect is outstandingly high.[2]

Experiences of the past five years reveal how remotely far we are from any miraculous software which will overnight transform our research libraries into "knowledge banks" capable of giving the right "answer" to any query, no matter how ineptly articulated.

A basic assumption is that we have a finite amount of human and cash resources. A further assumption is that people-costs are rising much faster than unit machine-costs, while the productivity of people in the library has hardly increased at all. A striking illustration of increased employee productivity in industry is documented in a recent *Fortune* article on the Toyota Motor Company of Japan.[3] Here is how productivity of Japanese automotive employees changed in twenty years:

1949: 1.5 cars/year/employee
1965: 20 cars/year/employee
1968: 28 cars/year/employee

We can demonstrate no such productivity increases in libraries; in fact, the opposite is likely to occur. As long as total productivity is a linear function of the number of employees, more staff and more supervisors will always be needed to control the ever-growing intake of publications and the spreading demand for library services. Correspondingly, the more costly a resource of declining productivity, the greater the net cost of a given output. Unit processing costs in the library must inevitably increase unless aided by machine processes. This fact and the universality of computers as a general purpose tool are the main force behind Dolby's contention that the computerization of cataloging is not only desirable but inevitable.[4]

The administrator considering an automation program faces two problems: (1) How does he allocate his limited resources? and (2) What can he do to in-

crease the productivity of his employees? Neither question can be answered without intimate knowledge of the present way of doing things.

Understanding the Present System

A prior requirement for any automation activity is a thoroughgoing, comprehensive analysis of systems and procedures currently in force. It is essential that analysis be carried out without any preconceptions or prejudices for or against automation. The determination of unit costs and unit times, as well as peak loads, are among the most needed data for making any kind of management decision on the desirability or feasibility of making changes. It may be discovered that change is neither desirable nor economical in certain areas. The conduct of a detailed analysis and evaluation is in itself a major task which in a large library can easily consume five to ten man-years.

A systems effort is a continuing activity, not something done once under the assumption that facts once obtained remain stable and fixed. Systems analysis is also a full-time activity; it is impossible for any staff person charged with operations to conduct systems analysis. What should be the ratio of systems analysts to employees? If there are two support staff for each professional, it is probably reasonable to have one full-time analyst for each fifty persons on the technical processing staff, inclusive of file-oriented functions, such as circulation. However, this is merely suggestive, not prescriptive.

Ideally, an analyst is a librarian, though excellent results can be obtained from a nonlibrarian if he meets the selection criteria described herein.

Extra-Library Assistance

The task of systems analysis and design cannot be delegated to any group of outside "experts." The users play the crucial role in both analysis and design. Technical people bring their own prejudices and a very imperfect knowledge of the mission and procedures of a given organization. In both business enterprise and in libraries, what *can* be delegated is programming and the technical methodology of implementing a system after management has decided *what* to do. Still, such contracted work will require constant monitoring by the systems analyst/designer and librarian to assure design integrity. A team approach dominated by a high regard for the quality of interpersonal relations is essential to success.

Where will the managerial talent come from to run an in-house development effort? If a library "grows its own," the orientation and educational task will be minimized. If one must go to the outside, how will he merge bibliographic and computing talent in the same person?

Suppose a library decides to delegate part or all of the technical (i.e., programming) task to an agency outside of the library. In this case, "outside the library" can just as well mean another agency on campus. A favorable political climate is a prerequisite to the success of this method. Affiliation with a local research project in information retrieval or with a scientific computing center is attractive, because there is a vast reservoir of intellectual talent—the scarcest resource in any automation task. However, such alliances can be biased equally by research interests on the one hand and implementation interests on the other. The intellectual challenges behind tough software problems are the stuff of life for the best people—the only people you really want working on hard problems. In trying to find the most efficient logical solutions, a programmer can easily be deflected from development aspects. A mutually challenging set of tasks should be determined which appeals to both sets of interests. To satisfy these conflicting

motivations takes a project manager with unusual catholicity of perspective.

Personnel Characteristics, Selection and Salaries

When one decides to mount an in-house automation effort, how does he get help with staffing? Judging the talent and performance capability of software people is not within our normal expertise. Even the experts have their troubles. Unless the librarian has learned a great deal about computing—possibly even learning some programming himself—he cannot reliably evaluate a candidate for systems analyst or programmer. You may recall that the ACLS report, *On Research Libraries,* closely parallels Orlicky's advice for businessmen, namely, that there is no alternative to the librarian's learning the computer art:

> Some programming experts must be brought into libraries but, more important, librarians must learn to use computers and must come to understand their strengths and limitations. This education process will take several years under the best conditions. From experience in other fields we can emphasize that there is no alternative to *library experts learning computation.* Any other course will lead to inferior results with great waste of money and effort.[5]

We have no contraindications to this advice. Librarians responsible for systems efforts must learn programming. We will nevertheless continue to depend upon knowledgeable people on campus, such as the staff of the scientific computation center or the administrative data processing center. Such assistance will be valuable, however, only in proportion to the degree to which the librarian is capable of making his goals understandable to these outsiders. About one man-year should be allotted for training and orienting each nonlibrarian sufficiently to ensure that the systems librarian can be confident that the details of a bibliographic application will be well understood.

It is common to divide systems development staff into at least three categories: systems programmers, applications programmers, and systems analyst/designers. Systems programmers work with and write the programs that offer to the applications programmers certain essential machine facilities, such as terminal access, special compilers, and languages for writing applications programs. The applications programmer writes the programs which actually execute user defined tasks, such as printing catalog cards from MARC tapes, creating overdue notices, issuing purchase orders, and the like. The analyst/designer is the person who gets right out with the users of the system and learns thoroughly their work. These categories of people are very different from each other; in most cases, it is not at all practicable to think about using them interchangeably.

Because he interacts directly with the user, the analyst/designer can make or break an application of automation. The analyst must be personable, patient, and respectful of the librarian's expertise. It is wise to be on guard against any candidate who appears abrasive, "smart alecky," likely to intimidate, or who feels that he already knows or can learn with little effort all that there is to be known about an application.

Programmers are different. Some are gregarious and sociable, others are loners. Most will prefer to work with the intellectual challenge of the application as described by an analyst rather than work directly with the user. Experienced programmers and users hardly ever speak the same language and can often misunderstand each other when they do get together. The qualified systems analyst knows enough about both worlds to be an effective go-between.

Systems development is expensive. A yardstick from industry indicates that it costs about $35,000 to support a systems programmer for one year. Suppose that he is paid $15,000; this might come to

some $2,000 per month inclusive of overhead, to which is added his requirements for machine test time. Machine time for testing might cost up to $1,500 per month, though he won't spend that much every month. His rate of expenditure will vary in accordance with task complexity and his own accuracy and efficiency, and, of course, in accordance with the pricing algorithm of the given installation. For library system development, I can cite one example. To develop the program for converting an incoming MARC tape to Stanford's local, internal processing format cost $8,000 in man and machine resources, inclusive of overhead. These are development costs, not operating expenses. The estimation of machine costs requires explicit information on specific program steps, machine configuration and pricing algorithms, the amount of execution time and utilization of other machine resources, the type of data being processed, and the general system complexity. An experienced cost accountant is really needed to interpret and break out the components of computer costs.

The salaries commanded by high quality analysts and programmers will come as no surprise. What may come as a shock is that one may have to pay more than one's own salary to recruit the right talent—this is particularly the case for large-scale applications which involve a degree of sophistication beyond normal batch processing. The larger the institution, the higher must these salaries be, because bigger organizations inevitably have more highly sophisticated computer services, which in turn require and attract higher-priced people. Sometimes one hears the complaint that there is a "shortage" of qualified computer people when what one really means is that someone does not wish to pay the salaries necessary to attract the desired people, or some institutional or legal constraint does not permit making the right offer.

Forecasting Whether and What to Automate

A thorough analysis of current systems should provide the administrator with information on the flow of material and data, the allocation of personnel, the organization, content, and use of files, and a complete inventory of forms. He will also obtain a profile of unit costs for various tasks within each library function of each subsystem. Working together with the systems analyst, the programmer, the staff of the computation center, and the policy makers of the university, the librarian should be able to identify not only the high unit-cost items and high total-cost items in this profile, but also those which are technically feasible for and readily susceptible to automation. There is no simple formula to define "readily susceptible to automation." This will be a function of the unique combination of machine and people resources present at a given institution and the director's own priorities based upon his program. It may be that some of the high unit-cost items cannot be aided by automation in the present state of the art; similarly, there could be a number of low-cost functions which occur in sufficient quantity to justify computer applications. For example, there is little doubt that circulation is one of the most profitable areas for exploration in any modern computer environment. But, at this time, the computer is likely to be of little immediate, economical aid to any intellectual task, e.g., original cataloging. A good text editing system, though, can simplify and speed the clerical aspects of copy preparation and card production.

There are three major, practical reasons for undertaking the automation of library functions: (1) to do something less expensively, more accurately, or more rapidly, (2) *to do something which can no longer* be done effectively in the manual system because of increased complexity or overwhelming volume of op-

erations, and (3) to perform some function which cannot now be performed in the manual system—providing always that the administrator actually wants to perform the new service, has the resources to pay for it, and is not endangering the performance of existing services for which there is an established demand.

In this connection, the mere capability of performing a given function by computer is not a sufficient reason for doing it. The technician is likely to believe "If we can do it, we should." The industrialist will assert "If we can make a profit, we should do it." The director of the library must decide whether that is the thing he really wishes to do in terms of his program, his budget, and his clientele.

With regard to the difficulties of library automation, an encouraging attitudinal shift is now evident. From a "trivial" problem, library automation has emerged as *the* intellectual challenge, rivaling information retrieval. There is, of course, always a substantial distance between the availability of a device and its actual application. Oettinger has outlined carefully the long, arduous struggles between conception of a device or technique, the building and testing of prototypes, and their emergence into production. In discussing the properties of educational devices, he cites the following factors to be considered in applying innovative resources: flexibility, generality, parallelism of access and simplicity of scheduling, quantity available, physical accessibility, reliability, ease of maintenance, degree of complexity, comfort for the user, and standardization. He also demolishes the idea (so glibly promoted by hardware salesmen) that possession of a device is synonymous with change of habit.[6]

Leaving aside practicality for a moment, a fourth justification for library automation activity is research—to learn whether certain new functions can be carried out with computer assistance, and if so, how to do them. The library community should neither abrogate nor delegate its research responsibilities. However, it is advantageous to integrate a variety of talents in complex computer applications. The stimulus of the nonlibrarian working together with librarians can aid in making sure that we do not suffer from "tunnel vision" and try merely to apply the new technology in the context of present limitations. These dangers are well delineated in SDC's report, *Technology and Libraries.*[7]

Facilities: Choice, Priority, Time

Ideally, the library ought to have under its own direct control all the necessary resources for complete system development. This is hardly ever achievable. The larger and the more complex the institution, the more likely is its computer facility to be complex and centralized.

Library use of computers suffers from two major handicaps: low priority for machine use and insufficient machine time. One way of dealing with these problems is to buy a significant interest in the machine. Another is for a number of neighboring institutions to form a consortium or processing center, or to utilize commercial services. Where the work load of a single library may not suffice to interest a computer facility, combined purchasing power may carry more weight. Still another method for overcoming problems of priority and time relies on prior, local political settlements, but in the end this method may not be the most efficacious. A problem-oriented solution is always better, and a great deal of work needs doing on the formulation of appropriate problem-oriented strategies for gaining computer support in library automation.

Libraries consume vast amounts of storage, use a lot of machine resources for input and output, such as keyboard-

ing purchase orders or collecting circulation transactions, printing catalog cards, and the like. Except for very complex software tasks, most library applications involve very little actual computing. They tend to be "input/output bound" rather than "processing bound," hence are much more closely allied to the operations of an administrative data processing center than to a scientific computing center. However, it would be misleading to suggest that this mere similarity in itself will be productive if much experimentation is involved. Oettinger points out:

> As many computer centers of all kinds have found out to their despair, routine scheduled administrative work and unpredictable experimental work coexist only very uneasily at best, and quite often to the serious detriment of both. Where the demands of administrative data processing and education require the same facilities at precisely the same time, the argument is invariably won by whoever pays the bills. Finances permitting, the loser sets up an independent installation.[8]

Turning to the scientific center, we see that its mission is to supply fast turnaround service to the research community. Its management generally does not look favorably upon file-oriented applications, because the machine overhead necessary to manage these functions detracts from the center's ability to service its clientele. Indeed, if there is any sophistication whatever in the scientific system, the number of competing users rises faster than the capacity of the system to meet demand, and this will certainly affect adversely any application not within that center's mission.

Dedicated or Shared Hardware

Dedicated hardware involves high fixed-costs of equipment and personnel; shared hardware involves high variable-costs for services performed, but if proper contracts have been negotiated, the user has some control over the kind and amount of services he purchases. A convenient feature of dedicated hardware is that the organization is beholden to no one—save the computer manufacturer, the telephone company, and the electric power company. But only one's own applications can be run on the machine and questions of priority and sufficiency of time do not exist; in fact, one may have time to sell. The type of machine one can have all to oneself will probably be a stripped-down model with a limited repertoire of software "smarts," and accommodating few peripheral devices.

A crew is needed to run the dedicated machine: operators to mount tapes, feed in decks of cards, and tear off and distribute printouts; systems programmers to maintain local software and keep the manufacturer's software and documentation up-to-date; and an administrator to schedule utilization and maintenance. It is also handy to have an external work load for idle time to help pay the rent. Also required will be backup arrangements so that operations can function on another facility during planned or unplanned downtime.

There are two very powerful arguments for not having one's own small computer:

1. First, a small taste of things one can do inevitably gives one an appetite for more sophisticated applications. The capability of the small machine is soon exhausted; changing to a larger computer may require a change in the operating system or programming language, either of which could require a lot of reprogramming.

2. The second reason is that small machines and small installations do not attract the intellectual talent needed to assure the most efficient use of machine resources. The better people naturally gravitate to the more sophisticated installations.

An alternative is to associate with a larger facility to take advantage of peripheral storage, special output devices

(such as terminals), and the scarce resource-talent. Because very little calculational computing is done in library applications, it may be possible to satisfy a local need with a "mini-computer" if the small machine can be used as a terminal for the central facility. But this kind of interconnection or networking is a substantial software task in itself.

Assuming that it is better to pay for service from a computation center, what kind of facility should be used? The differing job characteristics in scientific and administrative applications have already been mentioned. Administrative data processing is oriented towards "fixed field" applications, whereas library usage involves variable length records with many special graphic characters. Administrative applications are further characterized by large work loads which often require two or three shifts; their timing schedules are critical owing to payroll and tax calculations and the month-end loads imposed by the task of preparing budget statements for thousands of cost centers. Also, an administrative data processing center is generally about one software generation behind a scientific center, and its systems programs may not be as efficient. However, an administrative center is likely to be much more sensitive to matters of file security.

Ultimately, the research library looms as the largest continuous consumer of computer power on the campus. When that time comes, it is entirely conceivable that libraries may dominate the campus computer realm. No other agency on campus affords more intellectual interaction with the academic community than does the library—which is exactly the reason why it is important for large libraries to continue experimentation and research in library automation. A failure aggressively to push research on bibliographic applications could lead to second-class computer facilities, and could put libraries many years behind other research components of the academic community.

Networks

Demonstrations of electronic networking have now become routine, but it would be misleading to believe that the establishment of regular, error-free networking is just around the corner. Existing telephone networks were designed for voice communication, not data transmission. Much old equipment is still in use; even though electronic exchanges are being rapidly installed, it may be 1980 or later before new technology is fully applicable to land lines. And there are many companies competing for the production and marketing of "interconnect" equipment. Accompanying the juncture of telephone and nontelephone interests are issues of performance standards, reliability, and technical standards—including establishing national use of the new ANSI (formerly USASI) code for data interchange and telecommunication. It is also apparent that the problems of networking, even in the local environment, are of no small intellectual and technical depth, and it would be folly to imagine that a large number of independent local networks are going to interact successfully on the first try. In all, many technical and economic hurdles remain; a common hope of all educational users is the planned educational communication satellite which might assist in reducing transmission rates and increasing reliability.

Pricing Computer Services

Exactly how one prices centrally furnished services is at times a matter of conjecture, but invariably one of controversy. The overhead cost of running a center is considerable—two and one-half to three times the straight hardware rentals—and includes physical plant, hardware maintenance contracts, software maintenance, user education, large quantities of published documentation to pro-

cure and maintain salaries and staff benefits, insurance for equipment, electric power and air conditioning, full rental for equipment which may be only partially utilized, failsafe auxiliary power, paper, spare disc packs, telephone service, travel to computer conferences.

Any time a peripheral device is attached to a computer, there is an associated software overhead. Someone has to write the software which makes that extra device work within the local software environment. Since each environment is unique, the vendor's software is sometimes a square peg. That is why every computer center needs a ready supply of systems programmers and why additional devices require payment of a surcharge or "installation fee." But unlike a telephone installation charge, the "installation fees" for computer peripherals can never be one-time charges because computer systems are never static.

Computer resources are a very peculiar quantity. In any installation, there is a finite and measurable amount of computing power, although the users would like to behave as though there were an infinite amount of the resource. All pricing schemes are designed to ration the fixed resource in accord with the value of a particular service to a given user. Flexible pricing schemes have been devised to control the user's behavior in the hope of distributing equitably the costs of all available resources. Flexible pricing divorces pricing and costing, raising the price of some resources to pay for an associated resource, which if charged according to its true cost, would be prohibitively expensive for the user.

The good pricing algorithms recover total operating costs inclusive of overhead. Thus, in a large scientific center which may be terminal oriented and whose mission is scientific data processing, there are two actions which slow down service to the majority of users: mounting tapes and changing forms in the printers. To discourage these activities a special service charge may be imposed. Likewise, to balance the load on the machine (and to balance the budget), special low rates may be offered in the overnight service block or corresponding extra charges imposed during the day to run urgent jobs at a high priority.

One more observation: Although unit machine costs are going down all the time, the more one has of a cheap resource—like quick photocopying, for instance—the more one is likely to use it, and the net effect may be more money spent. It's the total expense that counts, not the unit cost. Consequently, the more facilities automation gives us, the more likely are we to need more resources rather than less.

Change As a Way of Life

One final question on the use of a central facility: What protection does the user have if the central facility decides to change its hardware or software? A change is only inconvenient for the transient research population but it is catastrophic for any continuing function like the library or the administrative data processing center. Written negotiations may offer some protection, but these tend to be political and are never as satisfactory as problem-oriented solutions. In any case, it would be self-deceptive to believe that any system design can be frozen forever. Hardware and software will change continuously; the rate of change needs to be controlled and stabilized.

Any major system change will affect forms, files, personnel allocation, procedures, and organizational structure. A change in any of these areas involves a training responsibility for the systems staff plus appropriate sensitivity to interpersonal relations.

Time Scale for System Development

What can one realistically expect concerning the time scale to design, install,

and operate an automated library subsystem, such as circulation or acquisition? Shoffner has pointed out that much early design work was based on the assumption—now known to be false—that existing library operations were already known in considerable detail.[9] To provide this detail in the form necessary for adequate system design work is very time-consuming, and the failure to realize the required time commitments is responsible for much of the slippage observed in current projects.

Contrary to popular belief, the design and installation of a computerized system to perform a given function is anything but a mechanical process. Yet too many people still imagine that it is a simple, straightforward process to flow chart an operation, write a program, and start running. Once a logically correct program has been written, it is true that its execution will be mechanical—if no equipment malfunction occurs. To be logically perfect, the program's intellectual design must account for every conceivable detail and alternative in the function being automated. This degree of perfection is hardly ever achieved the first time because of our lack of precise knowledge about our operations; it is here that we are confronted (rather brutally and expensively) by the conceptual error that everything there is to be known about a given library function is already known in complete detail. Programming bears a much closer resemblance to space exploration or to a large building construction project, where unanticipated problems are constantly intruding into well-laid plans.

To reiterate: No computer system can be implemented in the absence of a series of systematic prescriptions resolving in the minutest detail all possible alternatives for all possible actions associated with a given application. The designers must have an exact picture in advance of the extent to which these minutiae must be described, documented, and incorporated into a design before a single program can be written. Popular misconception still talks about "what the computer will do," whereas the programmer knows the computer will insistently and stubbornly do only the things it has been instructed to do. The man-machine gulf is deep enough so that every programmer wishes for an imaginary command or instruction for his computer: "Do what I mean, not what I said!" These remarks are not meant in disrespect of the research being conducted in artificial intelligence, simulation of the nervous system, and other advanced projects. The goals of those research projects are thousands, perhaps tens of thousands, of man-years away. Computers in a workaday, production environment must still be told everything or they will do nothing.

Transferability

Since the inception of the computer era, transferability of software and system design has been a recurrent hope and theme. Theoretically, a program once written to perform a specific function would not need to be written a second time by a second user. This hope has been dashed by four factors:

1. Scale of system complexity: The larger and more complex the system the more likely is it to have components that interact with the total system on that machine. There are relatively few problems with the transfer of single-purpose batch programs or program modules, but even here, if data are to be processed by the transferred programs, it is essential that the source data be identically formated and flagged. For this purpose, a translation program may be required. If there is much complexity to the data, less programming effort may be required to start from scratch. (In actual fact, the last statement may not be true, but the programmer will have to be convinced.)

2. Machine incompatibilities from one

computer manufacturer to another, and even within the same vendor's line of equipment, and a lack of "upward compatibility," a widely heralded feature which did not prove out in practice. Engineering changes made to new machines, if not incorporated into those already in the field, introduce incompatibilities within the same model of the same maker's computer.

3. A third deterrent to ready transferability is local alteration of the machine operating system and other system software, including use of different releases of compilers and languages. Changes in these "executive" programs, which run the computer or compile programs, can often make application programs inoperative. When system software is customized, subsystems no longer function as interchangeable parts on physically identical machines. It becomes a bit like trying to fit the door of a Chevrolet onto the body of a Plymouth; it can be done, but it isn't worth the effort.

4. The fourth source of difficulty for transference of designs and programs lies in the fact that library *X* rarely wants to do exactly what library *Y* wants to do. As long as we insist upon tailoring the bibliographic record to a real or imagined special local need, there is little likelihood that the programs which process MARC tapes at one installation can process them elsewhere. Only the acceptance without change of a centrally produced bibliographic record and the abandonment of customizing data to local requirements will enable system designers to begin thinking about one software package to work for many customers.

Access to Machine-Readable Files

The structural complexity of bibliographic records and the semantic ambiguities associated with them greatly complicate the access task wherever there is no direct human intervention. How, for instance, would a machine system distinguish the nearly 45,000 "London" entries in the Library of Congress Official Catalog? Even though these subtle intellectual problems have not yet been solved, experimental on-line systems have demonstrated great power to retrieve references with far greater speed and much more flexibility than is afforded by manual systems. But we lack long-term experience with many users employing a large data base. Further, the dependability of on-line files for bibliographic applications has yet to be demonstrated in a production environment, though I am reliably informed that the intellectual problems of file reliability and security may be near solution in several different commercial and military applications. The cost of maintaining in machine-readable form large files subject to immediate on-line access is prohibitive right now but could be within reach for technical processing applications of large libraries or groups of libraries, if economic and technical solutions can be found for networking problems. (Costs of file maintenance for large manual files in large libraries should be monitored continuously to see if a breakeven point is nearing which could justify selective change to machine-readable files with reasonable prospects of near-term payoff.)

Yet why keep available on-line very large files where the probability of access to a given item is exceedingly small? We know little about how users access bibliographic information from printed media and still less about how they might extract data from other kinds of files. Considerable prior experimentation with large, machine-readable files is necessary if we are to know how to organize those files. Even so, use of the files after a period of time might very well result in a continuously changing file structure, which, ideally, should be transparent to the user, i.e., apparent and meaningful only to the systems programmers. In the

future we probably need to be prepared for partitioned files with different technical designs and differing echelons of accessibility, ranging from printed media through on-line indexes and computer-output-microfilm (COM) files. Partitioned files might range from on-line indexes, through off-line book catalogs, special chronological or subject files, COM catalogs reissued yearly with current updates available from small on-line files. This variety of products and services can only be envisaged because of the richness of the MARC II format, which affords great selectivity of data elements for the user. By selective omissions, we can conceive of a graded series of files whose complexity and access time are inversely related, with appropriate cost trade-offs. New ideas for file organization and access are badly needed; all will have to be tested for economic and technical practicality and user acceptance.

The cheapest method of accessing a static, little-used file is via a printed medium. Interest in on-line library files has undoubtedly been stimulated by successful commercial applications, some of which parallel library uses. Airline and hotel reservation systems are good examples of successful on-line applications and rapid file accessibility. Both deal with an extremely perishable commodity. The same is true of industrial parts lists and inventory control systems, where orders and cash flow are the controlled items. How perishable is bibliographic information? A fairly good case can surely be made for the perishability of technical processing or circulation data, which by definition are high activity, "update-intensive" files. The case is likely to be somewhat weaker right now for low activity files. How much is the user willing to pay for immediate access to a file? Suppose one can satisfy 85 percent of the over-the-counter circulation queries by means of a batch system with 24-hour turnaround, as is the case at Columbia University?[10] What would be the value of a more sophisticated system which might reduce turnaround to an hour? To a minute? Suppose it costs twice as much to reduce turnaround to an hour? Five times as much to cut it to a few seconds? If automation is to be cost-effective, the resource must be appropriately matched to the task at hand.

But in this connection it is well to keep in mind the rapid development characteristic of computer technology. Fast-response, large-capacity storage devices may be available sooner than one imagines. What looks impossibly expensive or beyond reach today can easily become tomorrow's necessity. Continuous contact with technical developments is an indispensable part of the system librarian's responsibility.

Communication and Documentation

Without proper documentation, a job is not finished, and the systems analysis work, design, and programming are useless. There are five purposes to documentation:

To make progress visible to one's sponsor.

To communicate one's intellectual product in the absence of its creators.

To communicate designs—for staff knowledge and participation—from the moment of conception through all formal design steps, terminating in completely coded, working programs.

To record the reasons for specific logical decisions and design features so that the originator does not have to depend upon memory in the course of revising or debugging designs and programs.

To communicate project results to the outside world.

Documentation does not fall out automatically as a by-product of a system development effort; it requires rigorous discipline. Unfortunately, there is nothing inherently romantic or fascinating about report writing; it is a burden. The

provision of adequate documentation requires first a person who can write in clear, articulate English and who understands both computers and libraries. These people are expensive. A project with a staff of five professionals should at least consider having a full-time editor to relieve the principal investigator of extensive report writing. If the professional staff is ten or more, an editor is indispensable.

National Goals and Priorities

Institutional uniqueness is a characteristic of current library automation activity. Each library appears to be going its own way, applying automation in much the same fashion as it applies conventional methodology. There is little agreement on what to do, in what sequence it should be done, and how we should do it. In short, we lack a national plan for dealing with the intellectual and managerial problems of library automation efforts. Our current endeavors—save for establishment of the MARC II standard format—are as fragmented as the manual systems they are intended to replace. Do we want to create a series of incompatible, local efforts? How can we resolve the inevitable conflicts of interest among institutions of differing sizes, budgets, and academic programs represented within a given group?

Joseph Becker suggests that some form of "social engineering" is needed to make it easier for large research libraries to contract among themselves for major systems development. Of course, such a suggestion implies a far greater commitment to standardization than the library community has evidenced to date. It must be remembered that fundamentally libraries are in the communication business. Efficient communication is completely dependent upon standardization—a fact that is being focused by the machine's intolerance for ambiguity. When we leave our oldest and traditional "software"—natural language and the written word—to take up the electronic impulse, we enter a world of unforgiving, impersonal rigor. To make the change successfully, it is doubtful that we can continue to go our separate ways as we have so expensively done with cataloging and classification.

We have given up self-sufficiency in collection building; will we give up some local autonomy in technical processing to benefit from the economies of standardization? My fear is that if we do not, we shall have fewer and fewer resources remaining for service to our clientele. This is a special hazard as libraries—especially in private institutions—enter a period of increased budget visibility. By some means, the desired and needed national goals and priorities must be identified; if we do not do it, it may be done for us. Neither we nor our users may care for the results.

Conclusion

Librarians have succeeded in demonstrating that a variety of library technical processing and public service operations can be computer aided. Significant accomplishments have occurred with relatively modest investments. Though few institutions can yet directly utilize anyone else's efforts, we are probably no different from the computer world at large in this respect. Both worlds may be suffering from lack of a national plan. These national policy issues—standards, program priorities, the kinds and degrees of bibliographic access, and some concerted attack on the economic problems surrounding computer applications—still await solution. It is my conviction that the solution of these problems is essential in order that research libraries may be able to service the present and future requirements of their users. The full scope of the problems of library automation is just beginning to be realized. Now is the time to marshal the country's best brains and resources in response to the recommenda-

tions of the National Advisory Commission on Libraries. We are beginning to define the problem; that is significant progress.

References

1. Jacques Barzun, "New Librarian to the Rescue," *Library Journal* 94:3964 (1 Nov. 1969).
2. Joseph Orlicky, *The Successful Computer System: Its Planning, Development and Management in a Business Enterprise.* (New York: McGraw-Hill, 1969), p.21.
3. William Simon Rukeyser, "The World's Fastest-Growing Auto Company," *Fortune* v. 80, no. 7 (December 1969): 76-80 and 128-35.
4. James L. Dolby, V. J. Forsyth, and R. L. Resnikoff, *Computerized Library Catalogs: Their Growth, Cost, and Utility* (Cambridge: M.I.T. Press, 1969), p.16.
5. Max V. Mathews and W. Stanley Brown, "Research Libraries and the New Technology," *On Research Libraries* (Cambridge: M.I.T. Press, 1969), p.65.
6. Anthony G. Oettinger, *Run, Computer, Run.* (Cambridge: Harvard Univ. Press, 1969), p.44.
7. Carlos A. Cuadra, ed., *Technology and Libraries* (Santa Monica: System Development Corp., 1967). (Available as ERIC Document ED 022 481).
8. Oettinger, *Run, Computer, Run,* p.196.
9. Ralph M. Shoffner, "Economics of National Automation of Libraries," *Library Trends* v. 18 no. 4 (April 1970): 448-63.
10. Paul J. Fasana. (Personal communication to the author.)