Lawrence Auld
Head, Technical Processes
Oregon State University Library, Corvallis

PREVENTING FAILURE IN LIBRARY AUTOMATION

The original title of this paper was "An Equestrian Solution; or, Get a Horse!" but it was discarded because it would not KWIC index very well.

Cartoonists' views of automation are well appreciated by librarians, particularly one cartoon in which the computer center director appears rushing into the computer room crying to two assistants, "You're trying to program a fool-proof cross-reference system? Good Heavens, you'll ruin the computer!" Cartoonists offer their own remedies for the computer which fails to work, with suggestions ranging from a swift kick in the transistors to, as a last resort, assuming a humble posture on one's knees and offering prayers to the big gray boxes. One engineering student had another solution. When the computer refused to work for him, he started to take it apart.

Attempts to automate libraries encounter the same kind of difficulties as do attempts to automate other aspects of our lives; while some of these difficulties are soluble, others must be deemed failures. A failure by its very nature tends to be spectacular, but its prevention seems dull in comparison. Remedial action, usually drastic and appearing to the bewildered onlooker as a form of witchcraft, provides a bigger show; nevertheless, the focal point of this paper is the less spectacular one of preventing failure in library automation.

Instances of failure in library automation are not often discussed publicly, let alone published, for neither personal nor institutional pride favors a public display of disaffection. As a result library literature does not usually mention the possibility of failure except when an author sets out to prove that library automation is not possible. Friends in the computer center tell me that a similar reluctance to discuss failure exists in other areas of automation as well. This paper is an attempt to fill a void: to locate some of the principal

shoals on which library automation is likely to run aground and to chart a safe course.

The original plan was to develop an imaginary case study and from that to extract certain principles, but it soon became obvious that in such an approach, the case study itself would have assumed undue importance. As an alternate, the choice was made to identify certain types of failure which can occur, discuss the means of averting such failure, and suggest remedies for failure after the fact. Fictionalized and sometimes grossly exaggerated examples have been taken from an imaginary library at Chestnut College, a medium-sized school more or less a half century old and located not too far away. Please bear in mind that these examples are isolated and should be viewed as having little relation to each other.

Ten categories of library automation failure are identified in the approximate order in which they will occur. These are: (1) improper motivation, (2) poor communication between librarians and programmers, (3) faulty systems analysis and programming, (4) lack of feedback to systems analysts and programmers, (5) inflexible programs, (6) operator error, (7) inadequate machine capacity, (8) low priority for computer services to the library, (9) insufficient budgetary support, and (10) lack of long-term commitment on the part of top management. It may be noted that machine failure is not included in this list, for, while computers are subject to downtime, the occurence of pied information is rarely the fault of machinery, but rather the fault of people. The reliability of computers to do as instructed even when instructed wrongly is truly amazing. A programmer's lament might well be, "The computer always does exactly what it is told to do, damn it!"

The first step to successful automation is coherent and proper motivation for wanting to automate. The library that is moved to automate solely because there is equipment next door, because someone believes money can be saved, or because the Joneses have automated is going to have difficulty. Successful automation requires careful definition of goals without which there can be no clear-cut direction of action and therefore no clear-cut instruction in how to perform.

Let us take our first look at Chestnut College where, a few years ago, it was decided that the library should be automated. The rationale was simple: spare time was available on the small computer owned by the college, and other libraries were known to be automating. When Mr. Head Librarian received his orders, the only goals were to absorb computer time and keep up with the schools down the road. While this situation was not an act of failure itself, it was a gigantic leap into the primrose patch. At this point two things were possible for Chestnut College Library: it could stumble along and achieve very little (unless it be by accident), or it could stop for the moment and establish more precise and reasonable goals.

Once motivation has been properly directed and suitable goals established, the hard work begins; and this involves perhaps the most critical factor in the whole process of library automation, notably two-way communication between library people and computer people. Through communication, library people gain an understanding of the capacities and limitations of computer equipment, computer people gain an understanding of the strangely

complex variety of materials in libraries, requests and suggestions are put forward, decisions to accomodate or deny these requests are made, and out of this the philosophical basis of the system is established. If communication fails in this regard, the system subsequently designed will fail in many respects, at least from the library's point of view.

By now Chestnut College had decided to automate book order and circulation where two extremes of communication were immediately evident. Miss Acquisitions, aware that many aspects of her department's work were repetitive and could be simplified through data processing methods, had, on her own initiative some months earlier, begun to study the available literature to learn what she could about library automation. Thus, when Mr. First Systems Analyst arrived in her department (his first visit to a college library) a common basic vocabulary already existed. Mr. First Systems Analyst was bright, adaptable, and quick to learn the basic library vocabulary, so that the two were soon able to communicate effectively. Either could ask questions and could expect coherent answers. Out of this grew a clearly reasoned plan for a computer system of writing purchase orders, maintaining on-order files, affecting payments for books, and maintaining budget files. Miss Acquisitions generally understood the limitations within which she must work and was able to accept the omission of certain features for which she had hoped. Similarly, Mr. First Systems Analyst gradually modified his concept of a single straight line of action as he began to understand the vagaries of book dealers and their wares.

Contrast this with the situation in the Circulation Department, where Mrs. Circulation was actively hostile to learning anything about computers and Mr. Second Systems Analyst felt little need to learn about libraries. The two talked at each other, questions went unanswered, and explanations were not understood. Finally, a business-type system was set up which was scarcely compatible with handling books and certainly did not satisfy Mrs. Circulation's expectations with one exception—it did not work.

Although good communication is a prerequisite for good systems analysis, it does not follow that good systems analysis is necessarily a result of good communication. It is the systems analyst's responsibility to perceive the totality of the system on which he is working for, should he fail, the component parts may become independent sections which do not connect one with another. Even seeing the totality of the system is not enough, because it is also essential to establish the correct relationship of each part to all the other parts. Incorrect relationships assume different kinds of actions from correct relationships so that the results can be quite different. One test by which one can measure the degree of success or failure of a system is the extent to which the results vary from the original intentions.

At Chestnut College Mr. First Systems Analyst was unaware of the nature of serials so that the book order system on which he worked with Miss Acquisitions was designed solely for monographs. Because he was innocent of knowledge concerning serials and so-called near serials, assuming that one title is one book, he did not allow for the serial-like attributes of standing orders for certain monographic items. Thus, he designed the book order system so that upon payment for the first volume received on a purchase order, that

purchase order ultimately would be removed from the files. For a one-volume monograph or ten-volume set shipped and received on one invoice this was fine, but for a two-volume set with each volume shipped and billed separately, it failed to work because the system which he designed failed to take into account the proper relationship between monographs and would-be serials.

One may wonder how Miss Acquisitions met this limitation. Every separately shipped and billed item was finally treated as a separate purchase order even when originally ordered as only a part of one item. Each invoiced partial shipment was paid, the purchase order removed from the file, and a new purchase order for the remainder entered on the file. At the same timy the acquisitions staff had to be extremely careful not to make two payments on one purchase order, as the second payment would return the encumbrance to the outstanding balance a second time, creating an error in the budget file in that amount. Miss Acquisitions is looking forward eventually to have the book order system redesigned, so that she can order materials and not worry about partial shipments which presently endanger the integrity of the budget file.

While the systems analyst perceives, designs, and records the total system, the programmer is given the already-determined system within which he must construct the individual program segments which comprise the total system. The work of the programmer is directly dependent upon the quality of the system designed and given to him, but the fact that he is given a good system does not insure that he will write good programs. Although correct interfaces may be described in the system given to him, he may fail nevertheless to write programs which will satisfactorily mesh when these interfaces occur, or he may totally misunderstand the meaning of the system and write programs which in fact accomplish very different purposes.

At Chestnut College Mr. First Systems Analyst handed the completed book order systems work over to Mr. Programmer who was to write some of the programs. Mr. Programmer, neglecting to take the time to examine carefully the over-all system, began to work on a small section which involved updating files upon the receipt of books. With only a cursory glance at the record format, he wrote a program which would record the actual price of each item and then write an invoice-voucher. The program tested to his satisfaction and he turned it over as a completed piece of work.

During the first two production runs, only purchase orders were written and no receipts were recorded, but during the third week, receipts too were included. The program which up to this point had worked properly, suddenly produced wrong budget totals: the outstanding balance was lower than it should have been. After two days of searching, Mr. Systems Analyst discovered that Mr. Programmer had failed to understand a critical step in the payment sequence, in that he had failed to realize that when making a payment, he first had to disencumber the fund. The program he had written correctly subtracted each encumbrance from available funds and, as each book was received, also subtracted the actual price. Obviously, the outstanding balance was diminishing at twice the rate it should have been. Not only was it necessary to rewrite the program, but the budget also had to be recalculated. Three methods were possible: (1) because the system was very new, it was possible to back up, start over, and reprocess, correctly this time, all trans-

actions; (2) in an older situation it would have been necessary to calculate manually the correct totals; or, (3) a particularly clever programmer could have devised a special program to edit the files, but manipulating live data in this manner is risky although sometimes necessary.

Once a system is designed, programmed, and implemented, it is possible for the systems analyst and programmer to move to the next assignment leaving behind their indiscretions like progeny. This is bad not only for the analyst and the programmer who are deprived of knowledge of their effectiveness, but for the system as well, for there is no feedback with which to measure and improve its effectiveness. Even the best system will not be perfect because of inadvertent omissions or errors, and even if it were error free initially, outside changes would soon occur which would create problems, so that it would not suffice for long. Clearly, having a computer man on hand for periodic review and trouble-shooting is a good idea. He may be concentrating his attentions on another assignment, but he should be available for assistance when needed, and he should feel a sense of responsibility for the ongoing health and success of the system.

After Mrs. Circulation's retirement from her position at Chestnut College late in December, Miss Book Charge became head of the Circulation Department. From inherited records as well as her own memory she knew quite well about the difficulties which had been experienced the year before when the day-of-year passed 365 and began again with day-of-year one. All of the overdue books had then suddenly become due a full year later and several snags were still left over from that time. The small computer at Chestnut College had simply recorded the passage of time with the change in the day-of-year but without noting the change in year.

Accordingly, Miss Book Charge asked the computer center what could be done to avoid a recurrence of last year's failure. Both the program logic and the record format ruled out the possibility of inserting some identification for the year unless the circulation system were to be completely redesigned. Another possibility, the one finally chosen, was to continue numbering the day-of-year serially, as if there were no new year. Thus, what would have been day-of-year one in the new year now became day-of-year 366; February 1 became day-of-year 397; etc. This would tend to be confusing, of course even with a conversion chart, but books due on day 332 of one year would now continue to be due on day 332 of that year, and not a year later. Unanswered, however, was the question of what was to happen the following year when the day-of-year would pass 999 and would need four digits where only a three digit space was available.

A related point to be emphasized in this regard is that of the need for documentation. Both the systems analyst and programmer must document their work each step of the way. Then, should they drop dead or depart to another job, their work can be carried forward by someone else. At Chestnut College, for example, the pay checks did not arrive one week. The young man who had developed the computer programs for writing checks had gone to another job, and the advent of a new Social Security withholding rate could not be implemented because the programs lacked documentation. The Computer Center managed to get the checks out four days later with no thanks to the departed programmer.

Similar difficulties can occur when documentation in the form of operator instructions (about which more will be said later) and user instructions has not been provided as a part of the total systems package. Consider the problems that would occur at Chestnut College if the keypunch operators, who prepare input data for purchase orders and then prepare additional input data for updating the files and making payments, were expected to function with no written instructions. Data would stray from the record format boundaries, prices would be entered in wrong fields, wrong code numbers would be used, and staff turnover would bring in new keypunch operators who would find the lack of written instructions an even greater problem. A good systems analyst, by providing detailed written instructions, can prevent most of these problems.

Even a program with tight logic and a satisfactory product will fail when operating requirements exceed its flexibility. Anyone who has studied even elementary computer programming is aware of the need to be precise in setting up machine instructions, for the ability of the system to react is a direct result of the program structure.

At Chestnut College the effort was made to produce a book catalog for its branch campus library. This attempt failed because, although the formats were excellent, no provision was made for entering corrections, so that an error whether it be a single letter, a word, or a whole line, could not be manipulated separately. Errors once entered in the system could not be altered, because the programmer, attempting to protect the files throughout a series of sort and print routines, established a logic barrier to any changes in the files.

Closely related to program flexibility are the fail-safe and fail-soft concepts of system design. Presumably a fail-safe system will not fail because inherent safeguards have been designed into it. For instance, certain commercial installations actually rely upon a pair of computers, the second of which provides backup to the first, should it go down; but even a system such as this can fail if someone should pull the power plug on the second machine. Fail-safe is thus a relative concept, as we also know from at least one literary source.

In contrast, a fail-soft system can be likened to a soft emergency landing; in the event of failure, alternate backup procedures which are not normally a part of the system are available which will at least minimize the extent of failure. An example may be cited in the circulation system at Chestnut College, which was so designed that should the input equipment (of which there is only one set) go down, information identifying books and borrowers may be recorded by hand and keypunched later in the form of pseudo charge cards. If the system had been designed with automatic backup equipment as a integral part of normal operations, then it would have been a fail-safe system.

Of course, it is possible to design a system which is both fail-safe and fail-soft. An outstanding instance in which this was not done was the unsinkable Titanic designed as fail-safe and with inadequate fail-soft provisions such as a sufficient number of lifeboats. Libraries with limited budgets are probably well advised to favor fail-soft systems, which should offer less opportunity for

outright failure and certainly require fewer pieces of expensive equipment for backup.

The best systems analysis and programming will break down if the computer operator errs in his work. A well-developed system relies upon the computer operator for no decisions and only a minimum of interventions. Even a large computer will require some operator intervention in the form of loading input cards, initiating the execution of a series of programs, and loading the proper forms, while a smaller computer may require additional operator intervention as additional input cards must be loaded, when certain operations must be divided into one or more parts, and because forms may need to be changed more often because of limited capacity.

Miss Operator, a newcomer to the computer center at Chestnut College, was having difficulty running the book order programs; however, she did get through her task and sent the box of printouts back to the library. Within an hour Miss Acquisitions was on the phone with a budget problems. Although purchase orders totalling only $5,000 were processed that week, the outstanding balance had dropped by nearly $10,000. Also she had discovered that the purchase orders had different numbers from the Library of Congress card order slips and, in examining the master printout of all books on order, she found double entries for items but with two different purchase order numbers for each pair. Miss Acquisitions said that it would appear that Miss Operator had run the input cards through the computer more than once and thus had placed the items on file more than once. Why was this?

Miss Operator, not having read her instructions properly, had first loaded regular stock paper instead of purchase order forms into the line printer. For the remainder of the run she had managed to load the proper forms. At the end she realized her initial error, loaded the purchase order forms, and re-ran the input cards after which she assembled what she thought was a correct set of printouts and set them back to the library, all the time unaware that she had encumbered the book funds twice for this set of purchase orders.

Miss Acquisitions was then faced with going through the entire master list of books on order looking for instances of double entries for the same item and preparing a cancellation card for the member of each pair that was not represented by a valid purchase order card. A lot of time was wasted, but the budget file ended up with correct totals. A few weeks later a substitute operator ran the input cards through the computer twice, but with correct forms so that two purchase orders were issued for each item. Unfortunately, no one caught this mistake until the books began to arrive in pairs.

Operator error can also happen when insufficient documentation is provided, for without instructions, job stream, or forms listing, an operator can hardly be expected to execute a series of programs and not make a mistake. Operator error is disquieting at best. However, one can take comfort in that, in libraries, operator error is somewhat less than fatal. In the aircraft industry they call it pilot error.

A neatly designed system will work very smoothly so long as the data being processed do not exceed the capacity of the computer. But when its capacity is exceeded, what has been operating as a good system can suddenly cease to operate at all.

Returning to Chestnut College we find an order system which had worked successfully for over a year. A modest book budget, sufficient to generate purchase orders at about 80 per cent of the computer's capacity, suddenly was more than doubled. Accordingly, the computer, which was capable of handling only so many purchase orders in a given period of time, was now expected to handle something like 175 per cent of its actual capacity load. This failure may be viewed as a result of the inability of the machine to exceed certain limits and not as a result of faulty programming; however, in a very real sense, this was a failure in the original systems analysis, because an eventuality, perhaps a remote one at the time of the systems design, had occurred which was beyond the limits of the system.

Finding a workable solution was difficult, in any case. It was suggested that creation of a second, or parallel, system would double first the capacity, but this would have required either that all transactions be run against both systems (very expensive) or each transaction be identified as belonging to one of the systems. The latter would have required reprogramming and then would have been difficult to execute. Another suggestion was to revert to a completely manual system, but campus politics and pride prohibited such a move. The sudden installation of a much larger computer and reprogramming of the book order system could have solved the library's problem, of course, but Chestnut could not afford a new computer and, even if it could have, a year might well have elapsed before delivery.

Another solution which was proposed, and later adopted, was to continue the present automated book order system, run a parallel manual system for the overflow, but make all payments through the computer. For the manual system a new series of purchase order numbers was begun in which each number was prefixed with the letters CF (said by some to be the initials for "Computer Failure"). Here we may note, of course, that each of the proposed solutions would have violated the original system, and that the solution finally chosen was perhaps necessarily the worst in this respect.

A computer's time, just as an employee's time, must be scheduled so that important data to be processed can be submitted at the proper time, and so that more important work can be done ahead of less important work. Obviously, with such a schedule, the higher a job's priority, the sooner it will be done and the less likely it is to be set aside in favor of another job.

A clear understanding of the relative priority held by each of the library's production jobs within the whole of the computer center's work is essential. A general comment over coffee of "Don't worry, your work will get done" is not sufficient. A clear statement of specific priorities is needed so that the computer center staff can properly schedule the sequence of their work, and so that the library staff can depend upon the receipt of completed work at specified times. Without such a clear understanding, processed data will not be available for a scheduled staff to use in their work or for administrators to use as a basis for decisions.

The Chestnut College computer center neglected to establish clear priorities for the work it did for various departments including the library. As a result, during slack periods, everyone was happy, but as the computer center's

workload increased, more and more departments found themselves waiting for promised but not yet processed work. The library was among those whose waiting time was increasing. Circulation records, which were supposed to be run daily, were sometimes running two and three days behind schedule. By the time overdue notices were printed and mailed, their validity and subsequent utility were substantially reduced. One overdue run which was to be the basis for withholding grades for delinquent borrowers was delayed to beyond the time when the grades were released. The delay was said to be caused by printing of the address labels for next year's homecoming festivities—labels which would not be needed or used for at least six weeks. Thus, the priorities at Chestnut College were anything but clear and logical.

A computer is sustained with dollars. Although the route by which these dollars come to the computer makes little difference to the computer, it does make a great difference to some administrators. While Chestnut College has never operated its computer unless that time can be billed to someone, a nearby college makes no charges to campus users, for the latter's computer center is directly financed by the business office. Interestingly, the two centers do approximately the same amount of work in a year and require approximately the same number of dollars. In either case the ultimate ability to feed dollars to the computer rests with the total budget available to the school during the year. At Chestnut, departmental budgets are supposed to be inflated to a certain extent to permit the payment of computer charges, but the inflation is not always visible. At the neighboring institution these charges have been calculated in advance before departmental budgets are made.

At Chestnut College, say two years ago in March to further the illustration, Mr. Head Librarian was notified that he would soon be expected to automate certain functions in his library, and two persons from the computer center were assigned accordingly to work in the library. Head Librarian thought this was very nice until, at the end of the fiscal year, he was billed several thousand dollars for the salaries of the systems analyst and the programmer and for the test time on the computer. The prospect of paying an unexpected bill to the computer center for the following year as well was even less pleasant. Although his objections to the business office were not well received, a sort of compromise was reached: the bill for March through June was absorbed by the business office, but the succeeding year's bill was to be paid by the library. It happened to be a moderately stable fiscal year, so that by cutting library hours and trimming the book budget, it was possible to meet computer expenses. In a tighter budget year some even more drastic reductions would have been required—perhaps even dropping the computer from the payroll! In this instance the computer center, the business office, and the library may each be seen to have been at fault: the computer center for not explaining in advance what its billing policy was, the business office for not explaining anything until it was too late, and the library for not investigating the method of billing in advance.

The top management of an institution bears a double responsibility when it authorizes a new program or project. Not only has it committed current funds, but, if the project is to continue, it must commit future funds as well. A school's top administration may cheerfully authorize an investment

from this year's budget to automate some portion of the library's operations, but this is no kindness unless future support in succeeding years is committed at the same time.

Adjacent to Chestnut College is a community college where, five years ago, the administration said, "Let's automate!" Across the campus, massive efforts were made to convert to computerized processes. The library prepared a printed book catalog and the card catalog was phased out. Then, two years ago the school administration withdrew its support for further automation and severely reduced the budgets supporting present automation. As a result, the book catalog is now out of date, funds for a revised edition are not available, and how to proceed is a difficult question. Whatever solution is attempted will thus surely be expensive, and further shifts in administrative commitments to support programs may well make a bad situation even worse.

In this paper ten potential areas of failure in library automation have been identified and briefly described. Each can be prevented through a combination of foresight, knowledge, and understanding. From examining these types of failure, it may be remarked, in general, that work in library automation, to be successful, should reflect at least these ten important characteristics: (1) motivation based on clearly defined goals; (2) effective communication between librarians and computer people; (3) logical and complete analysis of the system, carefully executed programs, and abundant documentation; (4) constant feedback of the system's day-to-day operations to the systems analyst and programmer; (5) flexible and adaptable programs; (6) a competent computer operator; (7) computer machinery capable of performing the tasks required; (8) a high priority for computer services to the library; (9) adequate budgetary support; and (10) positive long-term commitment by top management, both now and in the future. With all ten of these, the chances for success are excellent.

In conclusion, I would like to draw a parallel which may not be unfamiliar. The kitchen garbage disposal unit is a familiar and generally accepted tool and is the object of both admiration and criticism. That it too can fail is evident from three common complaints: (1) it won't always work when overloaded or jammed; (2) it won't grind up tin cans and bottle caps; and (3) not every would-be plumber is qualified to fix it.