IMPLEMENTING MICROCOMPUTER SOFTWARE FOR MARINE SCIENCE LIBRARIES

Mary Katherine Politz
Librarian, LUMCON

In this age of computer, librarians are often called on to manage micro-computer systems, program generic software to library-specific applications or at the very least become conversant in computerese. Purchasing and implementing software for library use is one of the keys for successful micro-computer use in marine science libraries.

J.H. Katayama reminds us that "special librarians are often faced with micro-budgets, insufficient staff, and mini-collections, but with macro demands on service."¹ Maurice Freedman tells us that computer costs have decreased by 25 percent every single year since 1950 and personnel costs have increased by an even greater percentage in the same period.² These are the tip of the iceberg of library literature recommending computers as an answer to micro-resources and macro-demand. But selecting and implementing software seem to be major impasses in the use of computers for maximizing library services. The scenarios are familiar. After reading and researching, you have a micro-computer and software but you find the software doesn't work with the hardware or it does but you are so frustrated trying to make it work that you could scream. Or, after reading and researching, you have no clue as to how to decide which micro, which software and heaven forbid, which peripherals. The problems are related because the frustrations incurred in implementing software directly relate to the decision-making process of selecting software and hardware. Ann Wolpert jokingly writes

221
of hardware, software and "underware." "Underware" is her term for strategic planning - "the process of quantifying and understanding the systems that go behind the visible services - that makes for the successful implementation of a computerized library system." So we must better understand ourselves and our libraries as well as software in order to implement automated systems.

What makes micros different from the minis and the mainframes are the components (microprocessors-silicon chips), size, price and memory. Always remember that micros are smaller cheaper and have less memory. Software is a generic term describing all types of computer programs - it's what makes the computer go. There are many kinds of software, both library specific and generic. Some software is a non-programmers dream, a way to get computers to do things easily; others need almost a pro-grammer's expertise to get anything done.

Off-the-shelf packaged software ranges from simple to complicated: operating systems, specific single application packages, general application software, modular software, and programming language software. The operating system is a "software program comprising a series of functions responsible for the overall coordination and management of the hardware and software resources used by the computer system during processing." The operating system is as much a compatibility factor as the hardware. Single application software is inexpensive and designed to do one simple task easily and efficiently. These are the kinds of software that librarians dream of, but they are extremely inflexible. The formats have been predetermined and the user only types in information as directed by prompts. Game software, catalog card programs, recipe to grocery list converters, communication software, and mailing label compilers are exam-
pies. General application packages are large programs that do many simple everyday tasks. They are more expensive, because they are more complex; but, because of the universality of the tasks, they are very useful. Word processing programs and spreadsheet programs fit in this category. Modular software is usually a large expensive software package that can do many things and can often work with other software. Examples are database managers, integrated library software and integrated software like LOTUS 1,2,3. Finally, programming language software is often called a compiler and intermediates between the professional BASIC, COBOL, FORTRAN and PASCAL programmer and your computer. You, the novice, can use most of these kinds of software for everyday activities, such as balancing the budget; and you can manipulate some to do library specific tasks like creating bibliographies. You can also hire experienced programmers to use all of the above software to create whole systems of customized and very expensive software.

Library-specific software is a different issue entirely. It is available for many library functions and will be available for almost any library function, in time. Robert A. Walton divides software for libraries into successful, emerging and struggling application. The software that have proven successful and efficient are: word processing, accounting, numeric spreadsheets, card catalog production, database searching, computer aided instruction and mailing lists. Emerging applications are those that are being tested and are not 100% debugged. Into this category Walton places serials control, circulation control, acquisitions control and community information systems. Obviously librarians will be delighted when these are perfected. Struggling applications are the ones
we want the most: on-line public access catalog and cataloging. 5 But completely integrated systems are still in the future. Some features are part of most systems, some will be available within a year, some may be available someday. Because they are all created by different vendors for different equipment and using different programming languages, they will not all fit into an integrated system. This forces the librarian to choose one function over another. Explore the fact that non-library software may do the same task more easily and more efficiently. Choose library specific software very cautiously in this time of rapid change in software.

I find an inverse (or perverse) ratio with software. The simpler and less expensive the software, the easier it is to use, but it can only be used for one task. The more expensive, more flexible software, inversely, is more difficult and time-consuming to implement. Software seems to be proliferating faster than rabbits, so if a specific package is not available or compatible, wait a few days or weeks and it may soon be on the market.

The rule of thumb is to choose the software first to support specific library functions and then choose the hardware and peripherals to support the software. But before you make a decision, you must make some concrete plans and some hard decisions. Before purchasing software, you must first assess the library's needs, abilities and budget. Involve the staff in a thorough investigation of library functions and choose the ones most suited to computer strengths. Remember that micros are best used to streamline and speed internal procedures, especially repetitive tasks, in order to better serve the user. Be sure to prioritize these functions from necessary to optional. Assess the library-computer ambience as well.
Is there a comfortable private place to work with the computer where the printer noise will not disturb patrons or staff? Are there staff members with time, interest and some expertise, or do they all hate machines? A staff and needs assessment may clear up problems in the manual system that may eliminate the need for automation. And it may point out attitudinal problems, either antagonism or superenthusiasm, that can be dealt with before the computers arrive, thus easing the implementation process.

Try to be realistic in your assessment of staff and attitudes, your own included. If you or your staff have little or no computer expertise, but there is no lack of enthusiasm, begin with something small and easy. The satisfaction payoff is great with both a word processor and a spreadsheet system or a file system. This is also a good approach to win favor with apprehensive staff or upper management. If you need an acquisitions system to track monies and purchases, try a spreadsheet system instead of a database manager. If you plan to type several manuscripts with tables and bibliographies, type a few letters first. If all you need to do is help the cataloging typist create cards quickly, get a catalog card program for $200, not an online catalog unit for $500. In other words, don't tackle ambitious projects or critical library functions until you know what you are doing. Make sure the application is somewhat efficient on paper before you replace it with a computer. A micro can create order out of semi-chaos, but it cannot correct a totally chaotic system. In exactly the opposite vein, don't replace minis or mainframes with micros unless you are certain that the application warrants such a small system. Don't replace a simple, quick, well-functioning manual system with a micro.
Most library applications, however unique, can be implemented with off-the-shelf software. Customized software should be created only if the application is very complex and very unique. Generally, the most cost effective and useful software for our small marine science libraries is a word processing package, perhaps a spreadsheet package, a database manager and a communications package with a modem. You may want one bit of library specific software for a specific problem, like card catalog production or circulation. The database manager will be the most useful, because it is designed to manipulate files in many ways. Not only will it help with indexing our usual reprint and map collections, but with a word processor, it will also create overdue notices, catalog cards and order letters, and it may replace the spreadsheet package for budget applications.

Library literature abounds with articles about reviewing and researching computer literature, and interviewing vendors, colleagues and local users in an effort to choose appropriate software and hardware and I doubt seriously that I can add anything to this discussion. Firm written plans for your library needs; a realistic understanding of what you, your staff, and microcomputers can do; and a large amount of research and discussion of software and hardware will get you a good workable microsystem.

In any case, many of you have your own hardware and software already. You don't want to know that you've chosen the wrong system, you want to know how to make your system work. You, like me, perhaps, thought that choosing the system would be the hard part and plugging it in and getting it running would be simple. How wrong we were! I thought I could read the manual and get the computer to do things. Wrong! I thought I
could muck around and muddle through. Wrong, wrong, wrong! Now you need to invoke the keys to implementation: a plan, dedicated time, interest and colleagues.

Whether you do the implementation, or a staff member does it, and whether implementation includes learning to work a word processor or using BASIC to program a complicated library project or directing the input for a major database, establish policies and procedures. Decide who is to use the equipment and for what purpose. Where will the equipment be located and how will computer time be allotted? What tasks have highest priority? Will certain applications be implemented concurrently, or will one be implemented first? A multi-user system should have a manager for the mundane tasks of updates, paper acquisition, repairs and standards.

Try to get software manuals before the hardware arrives and read them thoroughly. If the manual is unintelligible, scour the markets and find trade books on your specific software and read them thoroughly. Sometimes software problems appear after the manufacturer's documentation has been published. Other problems are the result of misleading or incorrect documentation. These supplemental manuals explain these problems and also may explain special approaches to that specific software. Some software includes tutorials. These take time but are great learning aids. Some software comes with a learner's manual, a step-by-step guide to implementation. These, too, are great implementation aids and well worth the time.

Only after a fair amount of reading and understanding should you begin experimenting. Learn the system, practice, learn the most used commands, have fun with it! This is a means of building up confidence in yourself and the software. There will be problems and
frustrations, but when they are part of a play situation, solutions come easier. Don't try to implement a system alone. Join a user's group. Line up telephone numbers of colleagues, professors, vendors. Keep these close at hand and call the appropriate source at the least provocation. Ask everything from how to approach a programming problem to why one feature isn't working. It also helps to talk to colleagues working with the same software on similar machines. It is especially pleasant to work with another library staff member on similar problems. Beware the expert who can fix the problem more easily than he can explain what is wrong. The problem will be alleviated, but if it ever comes up again, you still won't know how to correct the problem. When you feel confident, begin to tackle the first priority library application. You will need to experiment with this as well. There will be many changes before the system is debugged, tested and corrected. Allow yourself or staff more time than you've anticipated - which is another way of saying that it will take much longer than you could possibly have imagined. Everything can be up and running sooner, if large blocks of time are allotted to the project, not an hour here or there. But limit time working with the micro to four hours, because after that point, working with the CRT will create eye strain and fatigue. Use utility programs like ProKey and Quickcode to facilitate programming modular and other software. They shortcut some operations and extend others and generally make for easier implementation. The interest factor is hard to define. You really have to want to implement a system to conquer the frustrations involved.

There are management tips that ease implementation. When choosing software, opt, if possible, for the kind that can be duplicated.
Originals should be kept separately in case of emergencies. Back-up copies are not essential, they are imperative. Every user should be responsible for their own disks. It should be drilled into every user's mind to make back-up copies of often used software, important data files and important programs after each session of input and to store the back-ups in a separate physical location. Copying disks works at the simplest level of computer operations and should be the first thing the user learns after learning to turn the machine on. With multi-user systems, it is good for each user to copy an update, so that the most recent update is always saved.

Label everything: disks, location of documentation, printer settings, parities, etc. Use notes in all your own programming and create documentation for your own systems, even if you are the only one who uses the system. Use menus and prompts often in systems designed for other users. Robert Walton says that "lack of reliable documentation will result in frustrated staff and patrons who will view the microcomputer as overly complicated, the staff as incompetent and the project as unsuccessful." Documentation is usually the last step of implementation and is often overlooked, but a fully documented, user-friendly system is much easier to use and to teach someone else.

There are two major enemies to the physical wellbeing of microcomputers and disks: electrical problems and dust. Beware of rainstorms, lightning, power surges and outages, and static electricity. All can damage hardware and disk memory. Surge arresters will help eliminate problems with power surges, if they are common in your area. Most computers require little in environmental ambience, but most do better with a dedicated power source. Don't load
up equipment on a single outlet. Use anti-static mats in offices with static-prone carpets. Store disks away from extremes of temperature, humidity and strong magnetic fields. Keep disks dust and smoke free. Discourage smoking and eating at the keyboard. Advice differs on cleaning dust from micro read-write heads. Some expect the user to do the cleaning, some require the cleaning by professionals. Check with your dealer about this.

Staff training is important and should not begin after all the programming, but at the beginning of the decision making process. Staff input throughout the planning and implementation will aid the implementers and end users. Change is difficult for many library workers, but if they are involved in creating and facilitating change, a sense of excitement and challenge will grow. Our small libraries are ideally adapted to implementation by several staff members, but larger libraries will have proportionately larger personnel problems.

Finally, give yourself and your staff a chance. The decision to automate a function may not have been as successful as hoped for, but then another function may improve with automation. Susan Epstein warns that "automation is not the right option for every library. Each library must decide whether or not any form of automation is the wisest use of its resources. It is not necessary to automate just because everyone else appears to be doing so. Library automation is changing rapidly. Decisions made today may, and probably will, be different tomorrow. The important point is that a library must continually reassess its position on automation and keep an open mind." 7 You must always remember the micro's limited storage capacity and slow response time with large databases. It is designed to be a small system for local application,
so it isn't really necessary to have it work well with other systems; it needs only to do the job allotted in your library. "When you decide to automate, know that a complete set of problems and costs come with this new technology," advises Maurice Freedman. And have no fear, change will be with us for a long time. As soon as you conquer DBase II, you will see the announcement for DBase III! But once the library staff and users find a taste of satisfaction in the change, they will demand more and more and the demand must be met. With an easy, efficient application, micros can save enough time to almost be considered another employee. But the time it takes to implement that application may take a staff member weeks or months to perfect. Knowing as much as possible about software and developing realistic expectations about what it can and cannot do can ease the implementation process from the very beginning. In fact, time, patience, knowledge and interest are the only weapons in the fight to tame software. For like the fox and the Little Prince, software and the user must tame each other. "'One only understands the things that one tames,' said the fox. 'Men have no more time to understand anything... If you want a friend, tame me...'

BIBLIOGRAPHY


5. Ibid. p. 58.


