A LIBRARY CATALOGUING SYSTEM USING MICROCOMPUTERS. ANALYSIS OF AN EXPERIENCE.

Ricardo Salas

Departamento de Ciencia de la Computacion
Facultad de Matematicas
Pontificia Universidad Catolica de Chile

The library system at the Pontificia Universidad Catolica de Chile has a project where the functions of Cataloguing, Circulation, Acquisitions and Retrieval are being implemented using several interconnected microcomputers, each supporting a different function. The experience gained while installing a multiuser online cataloguing system is discussed from the point of view of the applicability of commercially available resources . Conclusions are of interest to both implementors and those wanting to evaluate the cost of using powerfull microcomputers to install databases in a shared interactive environment.

INTRODUCTION.

University Libraries in developing countries do a significant amount of original cataloguing. This is due mainly to the domestic nature of their bibliographic material and to the absence of a network structure. In fact, the slow and costly cataloguing process usually becomes such a bottleneck, that maintaining an updated collection may be impossible without the help of computer equipment.

The kind of demands, characterized by the use of large files, online, with multiple indexing, that library systems put on time sharing facilities, result often in high costs and poor service. Thus, independence from large, centralised mainframes shared by a comunity of heterogenous users, is highly desirable.

Budgetary constraints and the emergence in the market of newer and fairly powerful microcomputer equipment, made us examine the alternative or designing a cataloguing system that would make use of their advantages. This was going to be a first stage of an integrated microcomputer library system. The academic environment was also an ideal home for it, given the experimental nature of the project.

This paper is intended as an analysis of the most relevant aspects of this experience, namely the specification of requirements, the selection of an operating system, machinery and software development tools, as well as a report of the problems encountered in the implementation process, and the solutions devised.

Trabajo presentado a The International Conference on the Application of Mini and Microcomputer in Information, Documentation and Libraries Tel.Aviv, Israel, March 13-18, 1983.

SPECIFICATION OF REQUIREMENTS.

In the system to be built is was deemed necessary to support four stations where the cataloguing personel would enter data to build and mantain a bibliographic catalog, and another station for doing supervisory work. In order to provide for compatibility with future systems as well as to facilitate the exchange of information, the MARC [1] format was chosen as standard layout for the data. The catalog was intended to house at least 50.000 entries.

The specification of a flexible user interface, that would allow cataloguers to enter and edit MARC records, came naturally as a first stage. Since the MARC format covers a very vast field, the definition of a practical subset to be used was studied by the cataloguing specialists. This selection had to be implemented in a way that would make easy to introduce changes in it, without compromising the existing catalog and the operational functionality of the system. Some existing systems [2,3,4] (mostlty based on large mainframes) where studied and discussed with the prospective users.

The cataloguer had to be able to perform searches in the catalog before attempting to enter a new record in order to avoid duplications and to save the effort of research that had been done in earlier entries. The nature of this searches was somewhat different than the searches one would expect from a terminal user (i.e. a student looking for bibliography), because the cataloguer is usually looking for a specific book.

Since the cataloguing sub-system was to be integrated with the other functions (circulation control, acquisitions and retrieval), provision had to be made to control data redundancy while keeping an adequate response time. The idea was to avoid the use of specially made hardware for economic reasons, and to facilitate portability. Networking hardware and software were not released at the time, so we had to rely on their preliminary specifications. To stay on the safe side, designs minimizing traffic across the nets were to be favored.

One thing that was clear from the beginning was that the equipment had to be able to support large (several megabytes) files on secondary storage and be able to handle them in a shared interactive environment.

OPERATING ENVIRONMENT.

The requirements, more that specifying equipment result in a specification for an operating system. They called in this case for a multiuser system capable of serving at least five or six users concurrently, without noticeable degradation of perrormance.

The classic multiuser system consists of a single CPU, lots of memory, and the apropriate number of I/O ports. The system

serves all the users by means of timesharing. An interesting alternative was using distributed processing involving multiple processors housed in the same cabinet. Here one of the processors acts as a master controlling the shared resources (basically disk I/O and printing), an the rest function independently. This type of parallel processing is usually called a tightly coupled network.

There were good reasons to believe that the programs were not going to be small or simple, and with the existing eigth bit equipment the use of a single CPU based operating system (MPM, Oasis, 8-bit Unix like, etc..) was likely to prove inadequate. Since interaction with data bases tend to define a system that is I/O bound, it was desirable to use an operating system with some sophistication in buffering management.

A very important point to consider was the fact that, because it was decided that the software was to be developed in house, a good set of software production tools was very much needed. No operating system could provide a wider choice than Digital Research's single-user CP/M. Most of this software can be used in some compatible multiuser operating systems.

With the above considerations in mind, a decision was made in the sense of selecting a multiuser operating system that would look like CP/M and support CP/M system calls, either directly or through emulation. Fortunately, one such system by the name of Turbodos was being released at the time by Software 2000 Inc. This operating system was originally built for a multimicrocomputer based on the S-100 bus and capable of handling up to 16 single card computers, each having their own 64K of RAM and their own console I/O.

HARDWARE.

The selection of a system is of course an iterative process, and the choice of a suitable operating system looked like a good place to start. Once that was done, particularly in this case of a multiuser micro system, the selection of hardware narrowed down to those microcomputers in the market that could run it. Today, there are like a dozen or so systems that can run Turbodos. In 1980 there was only one, namely an IMS 8000 made by Industrial Micro Systems. This system had among its options a hard-disk cartridge combination capable of holding almost 100 MB of storage, and could be configured as an eigh user system, each getting an enhanced CP/M environment with mechanisms for synchronizing concurrent interaction with shared files. Disk I/O operations were handled by the master computer where most of the distributed operating system lived. The master computer could use the remaining of its 64K RAM as buffers to increase speed. Once a program gets loaded into a user's private RAM, it can run asynchronously in that user's private CPU, and comunicate with her through its own serial port.

Although this operating system can handle a larger (24 bits) address space in disk than CP/M, compilers made for the latter will not generate the code to make use of this feature, and large files will have to be partitioned in chunks of 8 MB.

SOFTWARE.

Notably, the uniformity of the operating environment provided by the almost universal use of CP/M has resulted in such an enormous variety of editors, compilers, interpreters and data management packages, that the selection process becomes often a project in itself. In the populated world of CP/M software one can find some very professional tools as well as significantly less ambitious pieces of software, made probably with the hobbist in mind.

THE CHOICE OF A PROGRAMMING LANGUAGE.

Most applications in the microcomputer world are developed in BASIC. The most important reasons for it are perhaps the simplicity of the language and the existence of compilers and interpreters that are source compatible. Nevertheless BASIC is a language where systematic top down design does not result naturally, and maintenance of the software products written in it is far from convenient. The use of traditional languages like Fortran or Cobol, on the other hand, pays the additional price of a very slow development cycle without many advantages besides their obvious popularity.

Among the block structured languages, the obvious choices are Pascal, PL/1 and C. There are good compilers for all of them. Pascal has the advantage of being more popular, meaning that there is likely to be less time spent in training programmers to work in the project and a bonus when portability to other systems is an issue. On top of it there are more choices of compilers in Pascal than in PL/1 or C.

The Pascal compilers tested in our project where basically three: Sorcim's Pascal/M has an outstanding fast development cycle due to its good human engineering features and to the fact that it does not generate machine code but P-code,although it needs an interpreter. This fact conspires against speed of execution, and what is more important, a substantial part of the limited address space is used by the interpreter, leaving less room for programs and data base managers. Therefore, a technique of overlays has to be used generously if a sizable program is to be run. Pascal/Z, a product of Ithaca Intersystems, generates excellent and compact Z-80 code, but compiler diagnostics are poor and, at least in earlier versions, extensions for doing overlays where inexistent or awkward. We found a very good development tool in Pascal/MT+,which is now distributed by Digital Research. The compiler generates fast 8080 code, although the run time library is somewhat bulky. It supports modular compilation, and interfacing with machine code can be done easily. The development process -usually slow in true compilers- can be accelerated

by means of an optional speed programmimg package, consisting in a specialized editor that, among other things, checks your syntax errors without leaving the editor. The overlaying of modules is a little cumbersome in Pascal/MT+ since you have to supply addresses for the code often by trial end error. The editor holds the text in core, so for most programs you will be forced to decompose them in modules and link them together later, which may be a good programming practice.

We tried Whitesmith's Pascal which is basically a translator from Pascal to C. It lacks many of the extensions of other pascals, notably strings (which are obviously used a lot in library software) and relies for the rest of the compilation on the very good (but slow) Whitesmith C compiler. We did not find this system practical for a project where a fast development cycle was mandatory.

TEXT EDITORS.

They play a very important role in software development, both for writing programs and the documentation that goes with them. There are a lot of good choices here: In fact chances are that you will find much better editors for microcomputers than for your favorite mainframe. We already mentioned the Speed Programming Package. There are a number of other editors that would do a very good job. We use Micropro's Wordstar because its popularity (a lot of people know how to use it). It is easy to use and does a very good job as a word processor too. By the way, by having so much software around, we realized how damaging, in terms or productivity, can be to leave it in the system, giving the programmer so many choices. Once a selection is made, we advocate for the removal of the rest of the compilers, editors, etc., from the system or you will find yourself in a Babel tower in no time.

The tradeoffs between good human engineering features versus quality of the generated code is always present when compilers are evaluated. The size and overlay composition of text editors must be taken into account in the context of the multiuser operating system's I/O overhead and compilers capability of modular compilation.

DATA BASE MANAGEMENT SYSTEMS.

The Data Base Management Systems examined proved disapointing. They often lack the power to specify high level operations conveniently and most of them barely qualify as file indexing devices.

DBMSs are built either conforming to the relational or the hierarchical-network data models. The elegance and high level of the former makes implementation more of a challenge to the software house in terms of space economy and efficiency. That is probably the reason why the more comprehensive DBMS in the market are constructed using either the hierarchical or the network

model. But even those, when dealing with large files (tens of megabytes), either show gross operating inefficiencies or file size limitations that, overcomming would require a fair amount of that low level programming that we were trying to avoid in the first place.

The DBMS examined were basically two: MDBS supplied by ISE and Ashton Tate's DBaseII. DBaseII, which is probably a very good production tool in a less demanding project, was discarded because it lacked the ability to handle very large files and was not flexible enough to meet our preliminary design goals in terms of of user interfacing. We believe that this kind of tools are the ones to use in applications development, which allows you to dispense of compilers entirely. They have to get more sofisticated (and that is probably what is happening) if they are to be used in projects of this size.

MDBS (version 1) was a very attractive product, although somewhat expensive for microcomputer software standards. It is based on a network data model and claims to be able to manage files of gigantic proportions, variable length records and very complex structures. The truth is that as soon as our experimental database interfaced to a program - written in Pascal - grew to a size or a couple of megabytes, the tables that the system needed would not fit in memory. In trying too solve this problem, we engaged in a series of transcontinental phone calls with the manufacturers with very little results.

A problem common to all DBMS available at the time was their inability to deal with concurrent updating. They were clearly not meant to be used in a multiuser environment. A way of overcomming that particular problem is leaving all database interaction to one process and to comunicate with it through the net, a rather complicated solution.

We finally decided to go our own way as far as database management was concerned and wrote the data management routines in Pascal using hashed balanced trees [5]. We believe that it is beneficial to use vendor DBMS and we are constanlty evaluating newer products and releases hoping to replace our own package and decrease the size of the maintainable part of our systems (those parts of a system that are vendor packages are usually mantained by the manufacturer). Unfortunately, demo packages put restrictions that make difficult to evaluate efficiency and their ability to handle large files. Published reviews on the other hand, can be quite misleading. They rely strongly in manufacturers claims and are tested only superficially by reviewers. This is also true of reviewings of compilers, which are benchmarked with debugging aids set on some and not on others, resulting in unfair comparisons.

PROJECT MANAGEMENT.

This was a case where designing, implementing and testing in a top down fashion proved a very illuminating experience: The

user interface was developed first and, although no database management was behind it, it could be tried experimentally by the cataloguing personel. The improvements suggested by them where implemented shortly and the system submited to further testing. The resulting product was both satisfactory for the users and allowed for a better design of the data management portion of it. The modularity induced so naturally has made possible to make major modifications in either of them without having to change the other. Modules themselves are written as table driven automatons, which makes easy to adapt them to the constantly evolving cataloguing policies and standards.

The fact that a system is being developed in a microcomputer does not mean that project management can be minimized. In a multiuser system, tasks like backing up files, updating system software releases, etc., can be significant and require the attention of staff more trained than a regular mainframe operator. Our experience showed that leaving this chores to the programmers degraded their productivity severely.

FUTURE DEVELOPMENT.

The second module of circulation control is currently being finished. It features bar code readers to expedite transactions and has its own abridged version of the catalog. The sharing of data is to be implemented next through a gateway loosely connecting both networks to minimize data redundancy. The overall design considers the retrieval module to be the home of the catalog, since it will be there where most of the database activity will be concentrated. Network traffic will be diminished this way.

CONCLUSIONS.

The main source of economy in developing a multiuser microcomputer based cataloguing system is likely to be just the lower hardware cost. A wide choice of software development tools like editors and compilers of good quality can also be found at a very reasonable cost. Where vendor DBMSs prove incapable of managing large files adequately, data management utilities have to be developed from scratch resulting in a higher cost.

BIBLIOGRAPHY.

[1] Unites States. Library of Congress. Automated System Office, MARC formats for bibliographic data (Library of Congress, Washington 1980)

[2] OCLC, Inc., Cataloguing: User Manual (Columbus, Ohio: OCLC Inc., 1979)

[3] BALLOTS Center, Status Report on Ballots (Stanford University: Ballots Center 1978)

[4] Grosh, Audrey. Minicomputers In Libraries. (Knowledge Industry Publications White Plains, N.Y. 1979)

[5] Aurtenechea, F. and Figueroa H. Almacenamiento y Recuperacion de Informacionenarboles delta-AVL, Eng. Thesis, PUC 1977